Compartilhando Recursos
INICIATIVA NERDZONE

NerdZone

DOCUMENTAÇÃO DO PROGRAMA

Programa de Formação em Tecnologia

Agilidade e Gestão de Projetos · Atuar como Desenvolvedor · Desenvolvimento e Orquestração de IA

CARGA HORÁRIA TOTAL

68 horas

FORMATOS

Treinamento · Workshop · Oficina gratuita

EDIÇÃO

01 a 06 de setembro de 2025
29 de setembro a 04 de outubro de 2025

EMITIDO EM

novembro de 2026

MENTOR

Erick Prado
NerdZone Solutions

LOCAL

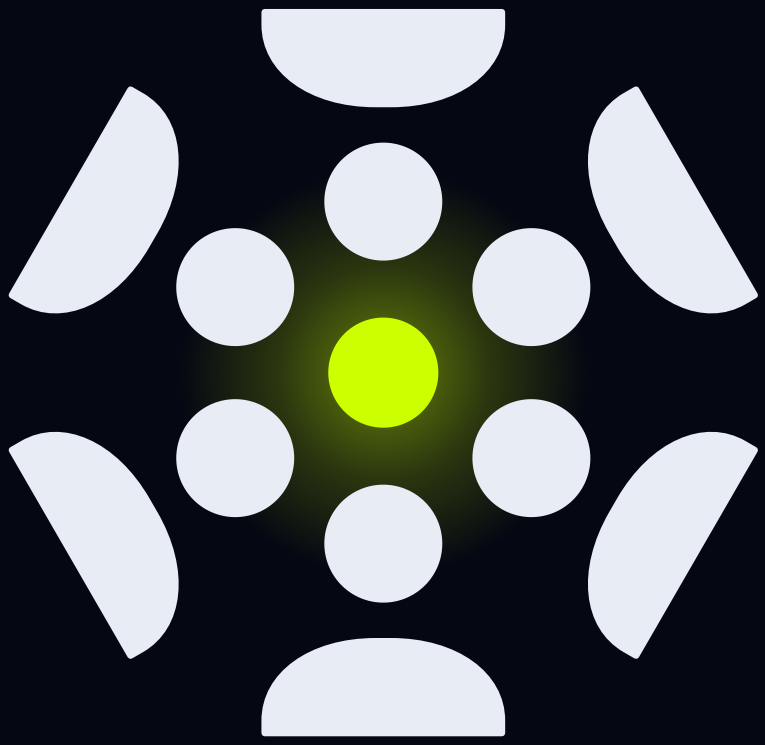
WeWork
Av. Paulista, 1374 · São Paulo, SP

PARCERIA E ORIENTAÇÃO

Estação Hack

SITE

nerdzonesolutions.com



Sumário

Sobre este documento 3

Participantes, formatos e modalidade 4

Visão geral do programa 5

Cronograma e carga horária 6

Metodologia 7

EIXO 01 **Agilidade e Gestão de Projetos** 8

Visão do eixo, objetivos e módulos 9

Competências, destaques e conceitos-chave 13

EIXO 02 **Atuar como Desenvolvedor: nível Estágio e Júnior** 17

Visão do eixo, objetivos e módulos 18

Competências, destaques e conceitos-chave 22

EIXO 03 **Desenvolvimento e Orquestração de IA** 26

Visão do eixo, objetivos e módulos 27

Competências, destaques e conceitos-chave 31

Fluxo integrado 35

Matriz de competências 36

Autoavaliação 37

Próximos passos 38

Quem conduziu 40

Sobre a NerdZone 41

Anexo A · Modelos 42

Anexo B · Checklists 46

Anexo C · Glossário 51

Sobre este documento

OBJETIVO

Este documento apresenta o Programa de Formação em Tecnologia da NerdZone: o que ele ensina, como se organiza e como levar o conteúdo para o trabalho de todos os dias. Serve como material de consulta e de estudo continuado.

PÚBLICO

Pessoas que estão começando na área de tecnologia e profissionais em nível de estágio ou júnior, que querem organizar o trabalho, construir software com qualidade e usar inteligência artificial com responsabilidade.

ORIENTAÇÃO DE LEITURA

Como usar este material

01 Comece pela visão geral e pelo fluxo integrado para entender como os três eixos se conectam.

02 Leia os eixos na ordem ou escolha o que mais faz falta agora. Cada um traz módulos, práticas propostas e checklists.

03 Use os checklists e os modelos dos anexos no trabalho real, adaptando ao padrão do seu time.

04 Volte à autoavaliação de tempos em tempos e escolha os critérios que ainda não são seus.

05 Consulte o glossário sempre que um termo novo aparecer.

QUEM E COMO

Participantes, formatos e modalidade

PARTICIPANTES

Turma presencial, com acompanhamento online; nomes preservados nesta versão pública.

FORMATOS

Treinamento **Workshop** **Oficina gratuita**

MODALIDADE

Híbrida (presencial, com acompanhamento online).

LOCAL DE REALIZAÇÃO

WeWork

Av. Paulista, 1374 · São Paulo, SP

PARCERIA E ORIENTAÇÃO

Estação Hack

Visão geral do programa

> **A tecnologia conecta, a gestão organiza, mas são as pessoas que fazem acontecer.**
>
> ERICK PRADO, NERDZONE SOLUTIONS

Tecnologia sem gestão vira improviso, e gestão sem tecnologia perde velocidade. O programa parte dessa ideia e reúne, em um só caminho, três frentes: organizar o trabalho, construir software com qualidade e acelerar a entrega com inteligência artificial (IA).

O Eixo 1 transforma uma necessidade em backlog, a lista ordenada do que precisa ser feito, e planeja o trabalho com Scrum (ciclos curtos chamados sprints) e Kanban (quadro que mostra o fluxo). O Eixo 2 leva o backlog ao código, com Git (histórico das mudanças, registradas em commits), testes e code review (revisão feita por outra pessoa). O Eixo 3 ensina a orquestrar a IA: conduzir assistentes de IA com contexto, regras de uso e métricas.

O material foi pensado para quem está começando. Cada tema explica o porquê antes do como, usa exemplos curtos e termina em uma prática proposta que gera um entregável concreto.

68 horas de carga horária total	**3** eixos temáticos conectados	**3** formatos: treinamento, workshop e oficina	**7** etapas no fluxo integrado, da descoberta à entrega	**1** prática proposta por módulo, com entregável

ORGANIZAR · CONSTRUIR · ACELERAR

A jornada em três eixos

EIXO 01

Organizar

Transformar ideias em backlog claro, com Scrum, Kanban, descoberta do problema (discovery) e estimativa.

EIXO 02

Construir

Entregar código com qualidade, usando Git, padrões, código limpo, testes e code review.

EIXO 03

Acelerar

Usar IA com contexto, regras claras de uso e métricas, sem abrir mão da responsabilidade humana.

Cronograma e carga horária

CARGA HORÁRIA POR BLOCO, EM HORAS



DIA	DATA	INÍCIO	
Bloco 1 · 01 a 06 de setembro de 2025			**35 h**
Segunda-feira	01/09/2025	19h	
Terça-feira	02/09/2025	19h	
Quarta-feira	03/09/2025	19h	
Quinta-feira	04/09/2025	19h	
Sexta-feira	05/09/2025	19h	
Sábado	06/09/2025	09h	
Bloco 2 · 29 de setembro a 04 de outubro de 2025			**23 h**
Segunda-feira	29/09/2025	19h	
Terça-feira	30/09/2025	19h	
Quarta-feira	01/10/2025	19h	
Quinta-feira	02/10/2025	19h	
Sexta-feira	03/10/2025	19h	
Sábado	04/10/2025	09h	
Atividades complementares · práticas dos participantes e revisão do mentor para aprovação, de 01/09 a 04/10/2025			**10 h**
Total do programa · 35 h + 23 h + 10 h = 68 h			

A carga horária é declarada por bloco. A distribuição por dia e os horários de término não foram registrados.

Metodologia

O programa combina explicação curta, demonstração e prática. Cada tema segue a mesma sequência: por que existe, como funciona, um exemplo pequeno e uma atividade para aplicar.

COMO O MATERIAL FUNCIONA

Princípios

01 **O porquê antes do como**
Antes de qualquer técnica, o material explica o problema que ela resolve. Quem entende o motivo adapta a prática ao contexto em vez de decorar passos.

02 **Fragmentos pequenos**
O trabalho avança em partes pequenas, fáceis de ler, testar e revisar. Isso vale para backlog, commits, revisões de código e pedidos ao assistente de IA.

03 **Prática com entregável**
Cada módulo propõe uma atividade com um produto concreto, como um item de backlog, um commit, um teste ou um checklist preenchido.

04 **Evidência antes de opinião**
Decisões se apoiam em algo observável: critério de aceite, teste, métrica ou revisão. Dizer que funciona só vale quando existe prova.

05 **Pessoas no centro**
Ferramentas e IA ampliam a capacidade, mas a responsabilidade pelo resultado continua sendo da pessoa e do time. Colaboração e comunicação fazem parte do método.

TREINAMENTO · WORKSHOP · OFICINA

Formatos das atividades

FORMATO 1

Treinamento

Sequência estruturada de exposição e exemplos, que apresenta conceitos, vocabulário e fluxos de cada eixo na ordem em que um apoia o outro.

FORMATO 2

Workshop

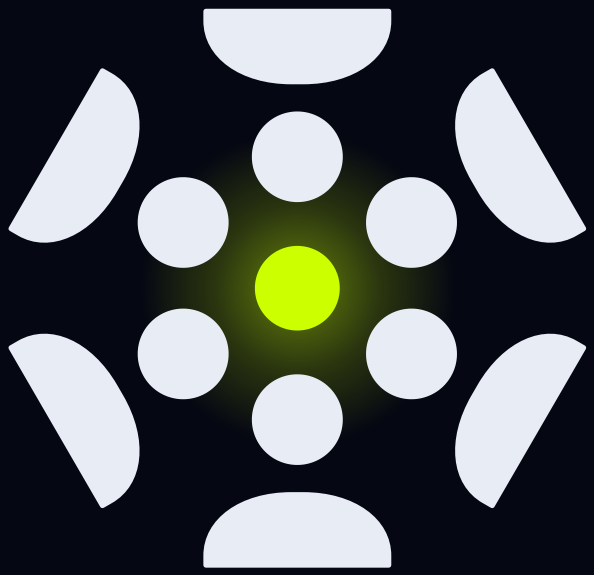
Encontro prático em que o grupo aplica o conteúdo a um caso que simula o trabalho real, com troca de ideias e feedback imediato.

FORMATO 3

Oficina

Oficina gratuita com tema focado e atividade curta, para conhecer o assunto na prática.

Compartilhando Recursos
INICIATIVA NERDZONE



01

EIXO 01 · ORGANIZAR

Agilidade e Gestão de Projetos

Organizar o trabalho para entregar valor cedo e aprender a cada ciclo.

EIXO 01 · ORGANIZAR

Agilidade e Gestão de Projetos

Este eixo mostra como transformar uma necessidade difusa em trabalho organizado, entregue em ciclos curtos e ajustado pelo que se aprende. Parte do Manifesto Ágil e do empirismo, aprofunda o Scrum conforme o Guia Scrum 2020, apresenta o Kanban e as métricas de fluxo, percorre o discovery de produto até o MVP e chega ao dia a dia do backlog, das histórias de usuário e dos critérios de aceite. Fecha com gestão de projetos, riscos, comunicação e saúde do time. A ideia central é simples: ágil não é um conjunto de reuniões, é uma forma de decidir com evidência.

EIXO 01 POR QUE IMPORTA

Quem começa na área costuma receber tarefas sem saber de onde vieram nem para que servem. Entender como o trabalho nasce, é priorizado, estimado e entregue ajuda a fazer perguntas melhores, evitar retrabalho e avisar cedo quando algo muda. Também prepara para diferentes times, porque os nomes das ferramentas mudam, mas a lógica de backlog, fluxo e feedback se repete.

EIXO 01

Objetivos de aprendizagem

- Explicar os valores e princípios do Manifesto Ágil e o papel do empirismo
- Descrever papéis, eventos e artefatos do Scrum conforme o Guia Scrum 2020
- Visualizar o fluxo de trabalho e usar limite de WIP e métricas de fluxo
- Percorrer o discovery de produto, do problema à hipótese validada e ao MVP
- Escrever histórias de usuário com critérios de aceite em BDD, Definition of Ready e Definition of Done
- Planejar com capacity realista, mapear riscos e comunicar-se com as partes interessadas

EIXO 01 · 7 MÓDULOS

Módulos

1.1 Fundamentos ágeis e Manifesto

Apresenta de onde vem o ágil e por que ele existe. Parte do Manifesto Ágil, seus quatro valores e doze princípios, e mostra Scrum, Kanban e XP como formas de aplicá-los. Introduz o empirismo: formular uma hipótese, experimentar, observar o resultado e ajustar. Mostra também que o primeiro valor, pessoas e interações, funciona como termômetro da cultura de uma organização, e apresenta o manifesto do time como documento vivo de acordos.

- Contexto: por que planos longos e fechados sofrem quando há muita incerteza
- Quatro valores do Manifesto Ágil, lidos como prioridades e não como proibições
- Doze princípios: entrega contínua, mudança bem-vinda, ritmo sustentável e simplicidade
- Empirismo: hipótese, experimento, observação e ajuste
- Ágil como guarda-chuva: Scrum, Kanban, XP e outras práticas
- Primeiro valor como termômetro cultural da organização
- Manifesto do time: acordos vivos sobre eventos, estimativa e convivência

PRÁTICA PROPOSTA

Montar o manifesto do time em uma página: propósito, duração da sprint, horário e duração dos eventos, forma de estimar, Definition of Ready, Definition of Done e regras de convivência. Cada acordo deve vir com a razão que o justifica e uma data para ser revisto.

ENTREGÁVEL

Manifesto do time de uma página, com acordos revisáveis.

1.2 Scrum na prática

Estuda o Scrum conforme o Guia Scrum 2020: um único Scrum Team com Scrum Master, Product Owner e Developers; os cinco eventos, sendo a sprint o contêiner dos outros quatro; e os três artefatos, cada um com seu compromisso (Product Goal, Sprint Goal e Definition of Done). Mostra onde materiais antigos usam outros termos, como Time de Desenvolvimento, e como ler a Review como inspeção e adaptação, não como aprovação.

- Valores do Scrum: comprometimento, coragem, foco, abertura e respeito
- Pilares do empirismo: transparência, inspeção e adaptação
- Scrum Team: Product Owner, Scrum Master e Developers, sem sub-times
- Sprint de um mês ou menos, com duração constante e cancelamento só pelo Product Owner
- Planning: por que a sprint é valiosa, o que pode ser feito e como será feito
- Daily de 15 minutos focada no Sprint Goal; as três perguntas são técnica opcional
- Review e Retrospectiva: inspeção do produto e inspeção do jeito de trabalhar
- Artefatos e compromissos: Product Goal, Sprint Goal e Definition of Done

PRÁTICA PROPOSTA

Simular uma sprint curta em papel: o Product Owner apresenta um Product Goal e um backlog, os Developers escolhem itens, o Scrum Team escreve o Sprint Goal e o grupo faz uma daily, uma review com demonstração e uma retrospectiva em três colunas, com ação e responsável.

ENTREGÁVEL

Sprint Goal, Sprint Backlog e plano de ação da retrospectiva, com responsáveis.

1.3 Kanban e fluxo

Apresenta o Kanban como método para visualizar e melhorar o fluxo de trabalho. Cobre o quadro com colunas e cartões, o sistema puxado, o limite de trabalho em andamento (WIP), as seis práticas do método e as métricas de fluxo: lead time, cycle time, throughput e diagrama de fluxo cumulativo. Fecha comparando Scrum e Kanban e mostrando como combinar os dois quando fizer sentido.

- Quadro Kanban: colunas como etapas do fluxo e cartões como itens de trabalho
- Sistema puxado: o time puxa o próximo item quando tem capacidade
- Limite de WIP e o custo de começar muita coisa ao mesmo tempo
- Seis práticas do método Kanban
- Lead time, cycle time e throughput (itens concluídos por período)
- Diagrama de fluxo cumulativo e como reconhecer gargalos
- Impedimentos e riscos como itens visíveis, com responsável e prazo
- Quando usar Scrum, Kanban ou os dois juntos

PRÁTICA PROPOSTA

Desenhar o fluxo real de um time, da ideia à entrega, em um quadro com colunas, critérios de entrada e saída e limite de WIP por coluna. Anotar as datas de início e fim de cinco itens fictícios e calcular o cycle time de cada um.

ENTREGÁVEL

Quadro com colunas, políticas explícitas, limites de WIP e cycle time calculado.

1.4 Discovery de produto: do problema ao MVP

Mostra como reduzir o risco de construir a coisa errada. Percorre as cinco etapas de discovery: alinhamento de objetivos com OKR, pesquisa, ideação, prototipação e refinamento até o MVP. Usa o duplo losango (primeiro acertar o problema, depois a solução) e a hipótese em formato de test card. Explica que o discovery pode vir antes do projeto, mas também pode ocorrer de forma contínua, em paralelo à entrega.

- Discovery e delivery: descobrir o que construir e construir direito
- Alinhamento de objetivos com OKR: objetivo e resultados-chave
- Pesquisa quantitativa e qualitativa: entrevistas, observação, analytics e testes A/B, que comparam duas versões
- Ideação sem pré-julgamento e geração de hipóteses
- Hipótese no formato test card: acreditamos, verificamos, medimos, estamos certos se
- Prototipação e teste com usuários reais
- MVP: a menor versão que gera aprendizado validado
- Duplo losango e discovery contínuo em paralelo à entrega

PRÁTICA PROPOSTA

Escolher um problema de um serviço do dia a dia, como agendar uma consulta. Definir um objetivo com resultados-chave, escrever duas hipóteses em formato de test card, desenhar um protótipo de papel e planejar um teste curto com uma pessoa que represente o usuário.

ENTREGÁVEL

Objetivo com resultados-chave, dois test cards e um protótipo de papel testável.

1.5 Do requisito à entrega

Acompanha uma ideia até virar trabalho pronto. Cobre a hierarquia do backlog (tema, épico, história, tarefa), a escrita de histórias de usuário com critérios de aceite em BDD, o checklist INVEST, as sete dimensões de um requisito, Definition of Ready e Definition of Done, refinamento, estimativa relativa e planejamento com capacity realista. Alerta para o escopo escondido, que estoura prazo e custo quando fica fora da conta.

- Hierarquia: tema, épico, história e tarefa, prática de gestão de backlog e não do Scrum
- História de usuário: como, quero, para
- Critérios de aceite em BDD: dado, quando, então
- INVEST: independente, negociável, valiosa, estimável, pequena e testável
- Sete dimensões de um requisito: ator, interface, ambiente, verbo, dado, regra de negócio e critérios
- Definition of Ready, prática do time fora do Guia Scrum, e Definition of Done, compromisso do incremento
- Refinamento contínuo, estimativa relativa em pontos (planning poker, votação com cartas) e capacity, o que o time consegue assumir na sprint
- Escopo escondido: documentação, pacote de deploy, deploy (publicar a versão), ambientes e suporte

PRÁTICA PROPOSTA

Pegar uma ideia curta, como cancelar um agendamento, e escrever uma história com título, descrição na voz do usuário e três critérios de aceite em BDD. Conferir com o INVEST, preencher as sete dimensões, estimar em pontos com um colega e revisar se ela cabe em uma sprint.

ENTREGÁVEL

História completa, com critérios em BDD, sete dimensões preenchidas e estimativa justificada.

1.6 Gestão de projetos na prática

Conecta o ágil à gestão de projetos. Trata escopo, prazo, custo e qualidade como restrições que se equilibram, o registro e o tratamento de riscos, o mapeamento de partes interessadas, o plano de comunicação, a abordagem híbrida quando a realidade pede e métricas úteis para decidir, como velocidade (quanto o time entrega por sprint) e burndown (trabalho restante ao longo do tempo). Inclui propósito e domínio do time e o squad canvas, que reúnem os acordos em uma página.

- Escopo, prazo, custo e qualidade: o que muda quando um deles muda
- Restrições levantadas na negociação que afetam custo e prazo
- Registro de riscos: probabilidade, impacto, responsável e resposta
- Impedimentos: item próprio, com responsável e prazo, acompanhado na daily
- Partes interessadas: mapa de interesse e influência e plano de comunicação
- Abordagem híbrida: planejamento por marcos com entrega iterativa
- Métricas úteis: velocidade, burndown, burn-up, lead time e planejado versus entregue
- Propósito do time, domínio do time e squad canvas

PRÁTICA PROPOSTA

Escrever o propósito do time (por que ele existe) e o domínio (o que pertence a ele) e preencher o squad canvas de um time fictício. Depois listar cinco riscos do trabalho desse time, com probabilidade, impacto, responsável e resposta escolhida.

ENTREGÁVEL

Squad canvas preenchido e registro com cinco riscos tratados.

1.7 Saúde do time e adoção ágil

Mostra como avaliar e evoluir a forma de trabalhar sem transformar isso em cobrança. Apresenta o radar de saúde ágil, um questionário em escala de 1 a 5 por dimensão, com gráfico de teia, e o plano evolutivo por sprint. Descreve o papel de um coach ágil, a adoção em fases e uma introdução ao ágil escalado, como o Scrum of Scrums para dependências entre times.

- Radar de saúde: dimensões, escala de 1 (nunca) a 5 (sempre) e gráfico de teia
- Dimensões típicas: time, planning, daily, review, retrospectiva, backlog, qualidade e cultura
- Usar o radar para conversar e priorizar melhorias, nunca para ranquear times
- Plano evolutivo: metas pequenas por sprint, como detalhar melhor as histórias, usar a capacity e reduzir bugs
- Adoção em fases: apresentação, primeira sprint, primeira review e retro, acordos e continuidade
- Papel do coach ágil: implantar, acompanhar, evangelizar e mostrar dados
- Dores comuns: dependências, priorização, padronização e resistência
- Ágil escalado em nível introdutório: Scrum of Scrums e demo integrada

PRÁTICA PROPOSTA

Aplicar o radar a si mesmo ou a um time fictício em três dimensões escolhidas, dando nota de 1 a 5 com um exemplo concreto para cada nota. Transformar as duas notas mais baixas em ações pequenas, com responsável e prazo, para revisar na rodada seguinte.

ENTREGÁVEL

Radar com três dimensões pontuadas e duas ações de melhoria com dono.

EIXO 01

Competências do eixo

Pensar de forma empírica
Formular hipóteses, experimentar, observar resultados e ajustar o plano com base em evidência.

Facilitar eventos ágeis
Conduzir planning, daily, review e retrospectiva com objetivo claro e tempo controlado.

Visualizar e gerir fluxo
Montar quadros, limitar trabalho em andamento e ler métricas de fluxo para achar gargalos.

Descobrir antes de construir
Transformar um problema em hipótese, protótipo e MVP com apoio de pesquisa.

Escrever e refinar requisitos
Redigir histórias com critérios de aceite em BDD e acordar Definition of Ready e Definition of Done.

Gerir riscos e comunicação
Planejar com capacity realista, tratar riscos e manter as partes interessadas informadas.

EIXO 01 · INSIGHTS, ALERTAS E DICAS

Em destaque

ALERTA

Termos de materiais antigos

Materiais antigos falam em “Time de Desenvolvimento”, chamam transparência, inspeção e adaptação de princípios e limitam a sprint a duas a quatro semanas. O Guia Scrum 2020 usa Developers, pilares do empirismo e sprints de um mês ou menos.

INSIGHT

Review não é aprovação

A Sprint Review é uma sessão de trabalho para inspecionar o resultado e adaptar o Product Backlog com quem tem interesse no produto. O aceite de itens é prática do time, não um portão do Scrum.

DICA

Pontos não são horas

Story points, prática de estimativa fora do Guia Scrum, medem esforço e complexidade relativos, não tempo. Uma tabela de pontos para horas só serve de partida para um time novo e deve dar lugar ao histórico real de entregas.

ALERTA

Escopo escondido

Ao negociar ou planejar, reserve tempo para documentação, pacote de deploy, deploy (publicar a versão), ambientes parecidos com produção, correções após a entrega e suporte. O que fica fora da estimativa vira atraso ou prejuízo.

EIXO 01

Conceitos-chave

Manifesto Ágil	Documento de 2001 com quatro valores e doze princípios para desenvolver software, que prioriza pessoas, software funcionando, colaboração com o cliente e resposta a mudanças.
Empirismo	Decidir com base no que se observa: formular hipótese, experimentar, inspecionar o resultado e adaptar. É a base do Scrum.
Scrum Team	Time único, sem sub-times, formado por um Scrum Master, um Product Owner e Developers (pessoas que criam o incremento: código, testes, design, análise). Entrega um incremento útil a cada sprint.
Product Owner	Pessoa responsável por maximizar o valor do produto e pelo Product Backlog, incluindo ordená-lo. Pode delegar tarefas, mas continua responsável.
Scrum Master	Responsável pela efetividade do Scrum Team. É um líder que serve: apoia o Product Owner, os Developers e a organização a praticar o Scrum.
Product Backlog e Sprint Backlog	Product Backlog é a lista ordenada do que pode melhorar o produto, cuidada pelo Product Owner. Sprint Backlog é o que o time escolhe fazer na sprint, com o plano para entregar.
Sprint	Ciclo de um mês ou menos, de duração constante, que contém todos os eventos do Scrum. Uma nova sprint começa logo após o fim da anterior.
Product Goal e Sprint Goal	Product Goal é o objetivo de longo prazo do produto e o compromisso do Product Backlog. Sprint Goal é o objetivo único da sprint, criado na planning, e o compromisso do Sprint Backlog.
Definition of Done	Descrição formal do estado em que um incremento, a versão utilizável do produto, atende ao padrão de qualidade exigido. É o compromisso do incremento e vale para todos os itens.
Definition of Ready	Acordo do time sobre o que um item precisa ter para entrar em uma sprint. É uma prática comum, mas não faz parte do Guia Scrum.
Limite de WIP	WIP (work in progress) é a quantidade de itens em andamento. Limitá-lo reduz espera e troca de contexto e deixa os gargalos visíveis.
Lead time e cycle time	Lead time vai do pedido à entrega. Cycle time vai do início do trabalho à conclusão. Ambos mostram previsibilidade e ajudam a achar gargalos.

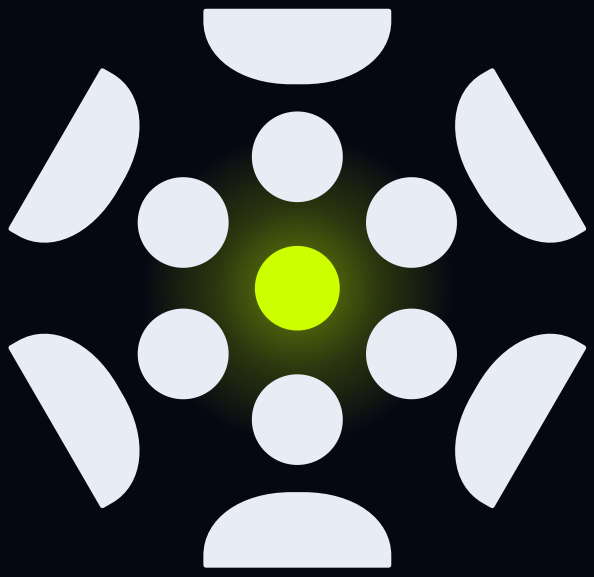
CONCEITOS-CHAVE (CONTINUAÇÃO)	
MVP	Produto minimamente viável: a menor versão que permite testar uma hipótese com usuários reais e gerar aprendizado validado antes de investir mais.
História de usuário e BDD	História descreve uma necessidade na voz do usuário (como, quero, para). BDD escreve os critérios de aceite como cenários: dado, quando, então.

EIXO 01 · VER ANEXOS

Modelos e checklists deste eixo

CHECKLIST 1-A	**História pronta para entrar na sprint**	Anexo B · p. 46
CHECKLIST 1-B	**Definition of Done do incremento**	Anexo B · p. 46
CHECKLIST 1-C	**Preparação da sprint**	Anexo B · p. 47
MODELO 1-D	**História de usuário com critérios de aceite**	Anexo A · p. 42
MODELO 1-E	**Test card de hipótese**	Anexo A · p. 42
MODELO 1-F	**Squad canvas em uma página**	Anexo A · p. 43

Compartilhando Recursos
INICIATIVA NERDZONE



02

EIXO 02 · CONSTRUIR

Atuar como Desenvolvedor: nível Estágio e Júnior

Construir com padrão, evidência e responsabilidade, uma mudança pequena por vez.

EIXO 02 · CONSTRUIR

Atuar como Desenvolvedor: nível Estágio e Júnior

Este eixo mostra como trabalhar em um time de desenvolvimento nos primeiros anos de carreira. O foco não é decorar uma linguagem, e sim adotar os hábitos que fazem o código ser aceito: entender a tarefa antes de escrever, seguir o padrão do projeto, entregar mudanças pequenas, provar com testes e conduzir o code review com maturidade. Cada módulo parte de uma situação real do dia a dia, explica o porquê antes do como e termina em uma prática proposta. O último módulo leva esses hábitos para a carreira: portfólio, perfil profissional, entrevistas e plano de crescimento.

EIXO 02 POR QUE IMPORTA

Quem está começando costuma achar que o problema é saber pouco de programação. No dia a dia, o que desgasta a confiança do time costuma ser falha de processo: código fora do padrão, teste que não foi executado, segredo enviado ao repositório, pull request enorme. Esses hábitos podem ser aprendidos desde o primeiro dia e valem em qualquer empresa, linguagem ou ferramenta. Quem os domina entrega com mais segurança e constrói a confiança do time.

EIXO 02

Objetivos de aprendizagem

- Adotar uma postura de aprendizado contínuo e pedir ajuda com contexto suficiente.
- Trabalhar com Git em branches, commits pequenos e pull requests claros.
- Mapear o padrão do projeto e verificar se a tarefa já está implementada antes de codar.
- Escrever código simples, com nomes claros, camadas separadas e validação no lugar certo.
- Provar a entrega com testes, build, lint e evidência registrada.
- Receber code review sem defensiva e documentar um projeto para portfólio e entrevistas.

EIXO 02 · 7 MÓDULOS

Módulos

2.1 Mentalidade e postura

Antes de qualquer ferramenta, vem a postura. O módulo trata de aprender a aprender com documentação oficial, experimentos pequenos e anotações, de pedir ajuda de um jeito que o outro consiga responder rápido e de se comunicar por escrito sem depender de conversa ao vivo. Apresenta também a ideia central do eixo: quem assina o pull request responde por ele, então a verificação começa em quem escreve.

- Aprender a aprender: documentação, experimento pequeno e anotação
- Responsabilidade e reputação de quem assina o pull request
- Pedir ajuda bem: contexto, o que foi tentado e resultado esperado
- Mensagem de erro completa em vez de “não funciona”
- Comunicação assíncrona: mensagem que se explica sozinha
- Avisar cedo sobre atraso e registrar bloqueios
- Receber orientação e aplicar sem repetir o erro

PRÁTICA PROPOSTA

Reescrever três mensagens vagas de pedido de ajuda, como “deu erro, pode ver?”, em mensagens completas com contexto, tentativa, erro e pergunta objetiva. Depois trocar com um colega e conferir se ele consegue responder sem fazer nenhuma pergunta de volta.

ENTREGÁVEL

Três pedidos de ajuda reescritos e um modelo próprio reutilizável.

2.2 Ambiente, Git e fluxo de trabalho

Git registra a história do código e permite que várias pessoas trabalhem ao mesmo tempo. O módulo mostra o fluxo completo: criar uma branch a partir da base atualizada, fazer commits pequenos com mensagens no padrão Conventional Commits, abrir o pull request e resolver conflitos com calma. O objetivo é que cada commit seja um passo que alguém consiga ler inteiro e desfazer sozinho se precisar.

- Repositório local e remoto, clone e atualização da base
- Branch por tarefa, com nome que remete ao item de trabalho
- Commit pequeno, com uma única preocupação
- Conventional Commits: tipos como feat, fix, docs, refactor, test e chore
- git status e git diff (ou git diff --staged) antes de cada commit
- .gitignore e arquivos que nunca entram no repositório
- Pull request: título, descrição e tamanho revisável
- Conflito de merge, quando duas mudanças alteram o mesmo trecho: ler os marcadores e decidir

PRÁTICA PROPOSTA

Em um repositório de treino, criar uma branch, fazer três commits pequenos no padrão Conventional Commits, provocar um conflito de propósito entre duas branches e resolvê-lo. Por fim, abrir um pull request com descrição completa para um colega revisar.

ENTREGÁVEL

Repositório de treino com histórico limpo e um pull request descrito.

2.3 Ler antes de escrever

Boa parte do trabalho de quem desenvolve é ler: a tarefa, a documentação e o código que já existe. O módulo ensina a transformar um item de trabalho (a tarefa registrada no backlog do time) em critérios de aceite testáveis, as condições objetivas para considerá-la pronta. Mostra como procurar arquivos semelhantes para seguir o padrão do projeto e como aplicar o gate "já está implementado?" antes de codar, evitando retrabalho e pull request vazio.

- Ler o item de trabalho: objetivo, critérios de aceite e fora de escopo
- O que a documentação já responde não vira pergunta
- Onde procurar: README, CONTRIBUTING, pastas e arquivos irmãos
- Mapear entrada, regra de negócio, persistência e testes semelhantes
- Gate "já está implementado?": buscar no código, nas branches e no histórico
- Comparar a branch atual com a base
- Quando parar e avisar em vez de codar
- Perguntar uma vez, de forma fechada (sim/não ou A/B), dizendo onde já procurou

PRÁTICA PROPOSTA

Receber uma tarefa fictícia de cadastro e, sem codar, entregar um mapa de uma página: critérios de aceite, três arquivos semelhantes encontrados no repositório de treino, o resultado do gate "já está implementado?" e as dúvidas que sobraram.

ENTREGÁVEL

Mapa de uma página com critérios, arquivos de referência e a decisão de implementar ou não.

2.4 Código limpo e simples

Código é lido muito mais vezes do que é escrito. O módulo apresenta o mínimo que faz diferença no dia a dia: nomes que revelam intenção, funções pequenas, camadas (partes do código com uma responsabilidade cada), validação no lugar certo e tratamento de erro que ajuda quem vai diagnosticar. Fecha com segurança básica: segredos, arquivos .env e dados pessoais não vão para o repositório nem para os logs. Refatorar é melhorar a estrutura sem mudar o comportamento.

- Nomes que explicam a intenção, sem abreviação obscura
- Funções pequenas, com uma única responsabilidade
- Camadas: entrada, regra de negócio e acesso a dados
- Seguir o padrão do projeto em vez da preferência pessoal
- Validação de formato na entrada e regra de negócio no serviço ou no domínio
- Tratamento de erro: não ignorar a exceção em silêncio nem vazar dado sensível
- Segredos e .env: variáveis de ambiente e arquivo de exemplo
- Dados pessoais e LGPD (Lei Geral de Proteção de Dados): coletar e registrar só o necessário

PRÁTICA PROPOSTA

Receber um trecho de código com uma função longa, nomes vagos, validação misturada à regra de negócio e uma senha fictícia escrita no código. Refatorar em passos pequenos, separar responsabilidades, mover o segredo para variável de ambiente e justificar cada mudança em uma linha.

ENTREGÁVEL

Código refatorado com diff pequeno e uma justificativa curta por mudança.

2.5 Testes, build e qualidade

Dizer que a mudança funciona não basta; é preciso mostrar. O módulo apresenta o teste unitário, o papel de build (montar o projeto), lint e verificação de tipos como gates antes do push e a forma de registrar a evidência: comando, resultado e versão testada. Apresenta ainda code smells, complexidade cognitiva e cobertura de testes como sinais para revisar o próprio código antes que outra pessoa o faça.

- Teste unitário: preparar, executar e verificar
- Testar comportamento e casos de borda (valores limite, vazios ou inválidos), não só o caminho feliz
- Build, lint e verificação de tipos antes do push
- Rodar os testes do escopo e registrar comando e resultado onde o time pede
- Diferença entre testado, recomendado e não executado
- Code smells: método longo, duplicação, acoplamento e nome ruim
- Complexidade cognitiva: por que condicionais aninhadas cansam quem lê
- Cobertura de testes: o que mede e o que não garante

PRÁTICA PROPOSTA

Dado um módulo pequeno com uma regra de desconto, escrever testes unitários para o caso comum e dois casos de borda, executar build e lint e montar uma nota de evidência com comando, resultado e o que não foi testado.

ENTREGÁVEL

Testes passando e nota de evidência com comando, resultado e limitações.

2.6 Code review

Code review é uma conversa técnica sobre o código, não sobre a pessoa. O módulo ensina a revisar o próprio diff antes de pedir revisão, a classificar cada comentário recebido (padrão, bug, preferência ou dúvida) e a responder de forma objetiva: o que mudou, por que alinha ao padrão e como foi validado. Mostra também como discordar com evidência, sem defensiva.

- Autochecagem: ler o próprio diff como se fosse de outra pessoa
- O que um revisor experiente procura nos primeiros minutos
- Pull request pequeno e com contexto costuma ser revisado mais rápido
- Classificar o comentário: padrão, bug, preferência ou dúvida de negócio
- Corrigir o mínimo necessário e atualizar os testes
- Responder com o que mudou, por que e como foi validado
- Discordar com evidência e sem tom defensivo
- Registrar o aprendizado para não repetir o apontamento

PRÁTICA PROPOSTA

Receber um pull request de treino com cinco comentários de revisão. Classificar cada um, aplicar a correção mínima e escrever uma resposta objetiva a cada comentário. Em seguida, revisar o pull request de um colega usando o checklist do eixo.

ENTREGÁVEL

Cinco comentários classificados e respondidos, e uma revisão feita para um colega.

2.7 Carreira e portfólio

Saber construir também é saber mostrar. O módulo parte de como se documenta um projeto completo: problema, objetivos, justificativa, requisitos, modelagem, cronograma e resultados esperados. Depois leva essa estrutura ao README do repositório, ao perfil profissional online e à preparação para entrevistas, e fecha com um plano de crescimento de júnior para pleno, com metas observáveis e revisão periódica.

- Estrutura de documentação: problema, objetivos, justificativa, requisitos, modelagem (desenho dos dados e de como se relacionam) e cronograma
- README que explica o problema, como rodar e as decisões tomadas
- Escolher poucos projetos e deixá-los bem acabados
- Perfil profissional online: título, resumo, projetos e links para o código
- Entrevista técnica: pensar em voz alta e perguntar quando faltar contexto
- Entrevista comportamental: método STAR (situação, tarefa, ação e resultado)
- De júnior para pleno: autonomia, escopo maior, qualidade e comunicação
- Plano de desenvolvimento com metas observáveis e revisão a cada ciclo

PRÁTICA PROPOSTA

Escolher um projeto próprio ou de treino e produzir três peças: um README completo, um texto de resumo para o perfil profissional e duas respostas no formato STAR. Fechar com um plano de crescimento de três metas observáveis para o próximo ciclo.

ENTREGÁVEL

README, texto de perfil, duas respostas STAR e plano de crescimento com três metas.

EIXO 02

Competências do eixo

Comunicação e pedido de ajuda

Escrever mensagens e descrições de pull request que se explicam sozinhas e pedir ajuda com contexto e pergunta objetiva.

Fluxo de trabalho com Git

Trabalhar com branches, commits pequenos e pull requests, resolvendo conflitos sem perder o histórico.

Leitura e aderência ao padrão

Entender a tarefa, mapear o padrão do projeto e confirmar o que já existe antes de implementar.

Código limpo e seguro

Escrever código simples, em camadas, com validação adequada e sem expor segredos ou dados pessoais.

Verificação e qualidade

Provar a entrega com testes, build, lint e evidência, e reconhecer code smells no próprio código.

Feedback e planejamento de carreira

Responder a code review sem defensiva e manter portfólio e plano de crescimento atualizados.

EIXO 02 · INSIGHTS, ALERTAS E DICAS

Em destaque

INSIGHT

Quem assina responde

O nome no pull request é o seu. Revisor e ferramentas ajudam a achar falhas, mas a responsabilidade de entregar algo correto e dentro do padrão é de quem escreveu.

ALERTA

Segredo enviado é segredo vazado

Se uma senha, token ou chave foi para o repositório, apagar o arquivo não basta, porque o histórico guarda tudo. Revogue e troque o segredo na hora, e avise o time.

DICA

Peça ajuda cedo, com o dever de casa feito

Defina um tempo para tentar sozinho, anote o que testou e então pergunte. Mensagem com erro completo, tentativas e pergunta objetiva costuma ser respondida mais rápido.

EIXO 02

Conceitos-chave

Branch	Linha paralela de desenvolvimento no Git. Permite alterar o código de uma tarefa sem afetar a linha principal até a mudança ser revisada e integrada à branch base, que a recebe.
Commit	Registro de um conjunto de alterações no histórico do Git, com uma mensagem que explica o que mudou. Cada commit deve tratar de uma única preocupação.
Conventional Commits	Convenção para mensagens de commit no formato tipo(escopo): descrição, com tipos como feat, fix, docs, refactor, test e chore. Facilita leitura do histórico e automações.
Pull request	Pedido para integrar uma branch à base. Reúne o diff (as linhas alteradas), a descrição e a conversa de revisão, e só é concluído quando o time aprova.

CONCEITOS-CHAVE (CONTINUAÇÃO)	
Code review	Revisão do código por outra pessoa antes da integração. Busca erros, inconsistência com o padrão do projeto e riscos, e também compartilha conhecimento no time.
Gate “já está implementado?”	Verificação feita antes de codar: buscar no código, nas branches e no histórico se o escopo da tarefa já existe. Evita retrabalho e pull request vazio.
Validação de entrada	Conferência de formato, obrigatoriedade e tamanho dos dados que chegam ao sistema, feita na camada de entrada. A regra de negócio fica em outra camada.
Variável de ambiente	Valor de configuração, como senha ou chave de acesso, definido fora do código. Costuma ser carregado de um arquivo .env que não vai para o Git; o repositório guarda só um exemplo sem valores reais.
Teste unitário	Teste automatizado de uma pequena parte do código, como uma função, isolada do resto. Prepara os dados, executa a parte testada e verifica o resultado esperado.
Linter	Ferramenta que analisa o código sem executá-lo e aponta erros prováveis e desvios de estilo, de acordo com regras configuradas no projeto.
Code smell	Sinal no código de que o desenho pode estar ruim, como método longo, duplicação, nome confuso ou acoplamento alto. Nem sempre é um bug, mas dificulta a manutenção.
Complexidade cognitiva	Medida de quão difícil é entender o fluxo de uma função. Condicionais aninhadas, laços e tratamentos de exceção aumentam o valor e cansam quem lê.

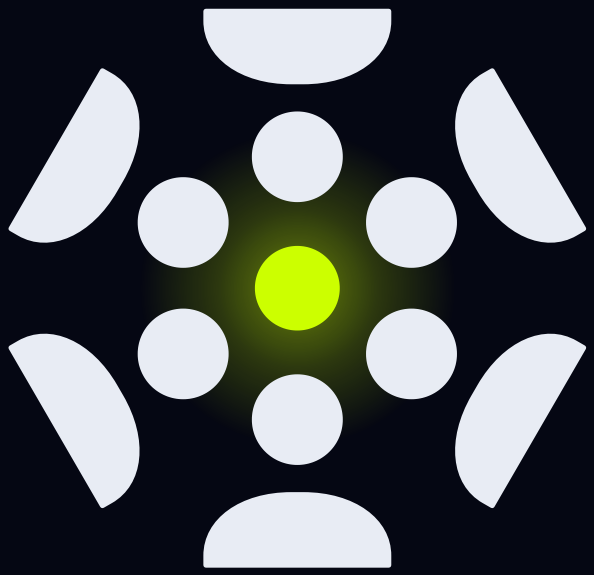
CONCEITOS-CHAVE (CONTINUAÇÃO)	
Cobertura de testes	Percentual do código executado pelos testes. Indica o que foi exercitado, mas não prova que o comportamento está correto nem que os casos de borda foram cobertos.
Método STAR	Estrutura para contar uma experiência em entrevista: situação, tarefa, ação e resultado. Mantém a resposta objetiva e baseada em fatos.

EIXO 02 · VER ANEXOS

Modelos e checklists deste eixo

CHECKLIST 2-A	**Antes do pull request**	Anexo B · p. 47
CHECKLIST 2-B	**Pedir ajuda bem**	Anexo B · p. 48
CHECKLIST 2-C	**Responder a um comentário de review**	Anexo B · p. 48
MODELO 2-D	**Descrição de pull request**	Anexo A · p. 43
MODELO 2-E	**Mensagem de pedido de ajuda**	Anexo A · p. 44
MODELO 2-F	**Resposta a comentário de review**	Anexo A · p. 44

Compartilhando Recursos
INICIATIVA NERDZONE



03

EIXO 03 · ACELERAR

Desenvolvimento e Orquestração de IA

A IA acelera quem sabe dirigir: contexto, revisão e prova em cada entrega.

EIXO 03 · ACELERAR

Desenvolvimento e Orquestração de IA

Este eixo mostra como usar inteligência artificial no desenvolvimento de software de forma segura, produtiva e verificável. A IA funciona como copiloto técnico: propõe, resume e rascunha, mas quem decide, revisa e assina a entrega é uma pessoa. O percurso começa pelos princípios e pela engenharia de contexto, passa por casos de uso e governança, chega à orquestração de agentes, instruções persistentes e gates de qualidade, e termina com métricas para evoluir com evidência. Tudo é ensinado com práticas curtas, modelos reutilizáveis e checklists que podem ser adaptados a qualquer ferramenta ou empresa.

EIXO 03 POR QUE IMPORTA

Quem está começando tem acesso a assistentes que escrevem código em segundos. O risco é aceitar o que parece certo sem entender. Aprender contexto, revisão e prova transforma a IA em alavanca de aprendizado, e não em atalho frágil. Esse hábito também gera confiança: time e liderança confiam em quem entrega código correto, consistente e revisável, com ou sem apoio de IA. Ferramentas mudam rápido, o método permanece.

EIXO 03

Objetivos de aprendizagem

- Explicar por que a IA é copiloto e quais decisões continuam humanas
- Escrever prompts com contexto, papel, tarefa, restrições, critérios de aceite e formato
- Priorizar casos de uso pela relação entre valor esperado e risco
- Aplicar o checklist de governança antes de enviar qualquer dado a uma ferramenta de IA
- Orquestrar contexto, agentes, revisão e gates até uma entrega rastreável
- Medir o efeito da IA com baseline e registrar experimentos

EIXO 03 · 7 MÓDULOS

Módulos

3.1 IA como copiloto, não piloto automático

Apresenta o que a IA generativa faz bem e onde falha. O modelo produz texto plausível, não garante verdade. Por isso a pessoa mantém arquitetura, segurança, decisão técnica, revisão e teste. O módulo explica alucinação, dependência excessiva e o princípio da responsabilidade: quem assina a entrega responde por ela, independentemente de quem rascunhou o código.

- O que é IA generativa e como ela produz respostas
- Copiloto e piloto automático: quem decide
- Alucinação técnica: a API ou regra que não existe
- Humano no controle: revisão, teste e decisão final
- Arquitetura antes do prompt
- Dependência excessiva e perda de pensamento crítico
- Responsabilidade de quem assina a entrega

PRÁTICA PROPOSTA

Dividir um pequeno problema de código em duas partes: o que a IA pode rascunhar e o que exige decisão humana. Pedir um rascunho a um assistente de IA, localizar uma afirmação duvidosa e confirmá-la na documentação oficial.

ENTREGÁVEL

Quadro com decisões humanas, trechos delegados e uma afirmação verificada na fonte oficial.

3.2 Engenharia de contexto e prompts

Prompt bom não é frase mágica, é engenharia de contexto. O módulo ensina os seis blocos de um pedido: contexto, papel, tarefa, restrições, critérios de aceite e formato. Mostra como fornecer exemplos e trechos do projeto e como guardar prompts em modelos reutilizáveis, para que o time melhore junto em vez de começar do zero.

- Contexto: sistema, domínio, stack (tecnologias usadas) e problema
- Papel: desenvolvedor sênior, arquiteto, QA ou revisor
- Tarefa descrita com precisão
- Restrições: padrões, camadas e tecnologias permitidas
- Critérios de aceite da resposta
- Formato de saída: tabela, checklist, código ou registro de decisão
- Iterar: pedir explicação e corrigir o rumo
- Biblioteca de prompts reutilizáveis

PRÁTICA PROPOSTA

Reescrever um pedido vago, como “faça um login”, usando os seis blocos. Executar as duas versões em um assistente de IA, comparar as respostas e anotar o que cada bloco mudou no resultado.

ENTREGÁVEL

Prompt vago e prompt estruturado lado a lado, com a comparação das respostas.

3.3 Casos de uso em engenharia e matriz valor x risco

Percorre seis frentes em que a IA ajuda o ciclo de desenvolvimento: código, legado, documentação, requisitos, testes e incidentes. Cada caso é avaliado por valor esperado e risco, o que orienta por onde começar. Documentação e cenários de teste vêm primeiro; dados sensíveis e treinamento de modelos exigem governança.

- Aceleração de código: esboços, refatoração local e scripts
- Legado: explicar regras e criar testes de caracterização
- Documentação: ADR (registro de decisão), runbook (guia de operação) e guia de endpoint
- Requisitos: regras de negócio, lacunas e perguntas abertas
- Testes: cenários em formato Dado, Quando, Então (BDD), dados fictícios e casos de borda
- Incidentes: hipóteses de causa e plano de validação
- Matriz de valor e risco para priorizar
- Critério de aceite para código gerado por IA

PRÁTICA PROPOSTA

Listar cinco tarefas reais da sua rotina de estudo ou trabalho e posicioná-las na matriz de valor e risco. Escolher a de maior valor e menor risco e redigir a hipótese de um pequeno experimento.

ENTREGÁVEL

Matriz preenchida e hipótese de experimento escrita em uma frase.

3.4 Governança, segurança e uso responsável

Define o que pode e o que não pode ser enviado a uma ferramenta de IA. Cobre dados pessoais e a LGPD (Lei Geral de Proteção de Dados), segredos, código proprietário, contratos e licenças, além de critérios para escolher ferramentas: segurança, integração ao fluxo de trabalho, qualidade, administração, custo e portabilidade. O foco é criar o hábito de perguntar antes de colar.

- Dados que não vão para ferramentas sem aprovação
- Segredos e credenciais: nunca no pedido, e troca imediata se vazarem
- Dados pessoais e a LGPD
- Retenção de dados e política de treino da ferramenta
- Licenças e propriedade intelectual do código gerado
- Mascaramento e uso de dados fictícios
- Critérios para escolher ferramentas
- Registro de uso para auditoria e aprendizado

PRÁTICA PROPOSTA

Analisar um trecho de código de exemplo que contém um token, um e-mail e o nome de um cliente, todos fictícios. Identificar cada dado sensível, mascarar o trecho e só então formular o pedido ao assistente de IA.

ENTREGÁVEL

Trecho mascarado e checklist de uso preenchido com a decisão de enviar ou não.

3.5 Orquestração: do contexto à entrega

Orquestrar é organizar contexto, papéis e verificações em torno da IA. O módulo cobre instruções persistentes do projeto, playbooks reutilizáveis, vários agentes especializados em paralelo com permissões limitadas, verificação adversarial por um segundo agente, gates de build e qualidade, evidências e o respeito à política de autoria da organização.

- Instruções persistentes: padrões do projeto sempre no contexto
- Playbooks reutilizáveis por tipo de tarefa
- Agentes especializados em paralelo, com escopos separados
- Limites do agente: permissões mínimas e confirmação humana
- Fragmentos pequenos, revisáveis e com commit próprio
- Verificação adversarial: um agente tenta quebrar o resultado
- Gates: build, testes, lint e análise estática
- Evidências e política de autoria da organização

PRÁTICA PROPOSTA

Escrever um arquivo curto de instruções do projeto com stack, organização de pastas, regras de teste e itens proibidos. Dividir uma tarefa em dois papéis, implementação e revisão, e comparar o que cada um encontra.

ENTREGÁVEL

Arquivo de instruções do projeto e relato curto da divisão entre implementação e revisão.

3.6 Medir e evoluir

Adoção de IA precisa provar valor. O módulo ensina a definir baseline, hipótese e métrica antes do experimento, comparar tempo, qualidade e retrabalho e decidir entre escalar, ajustar ou descartar. Inclui rituais de aprendizado compartilhado, como revisão de prompts e demonstração de resultados, e o registro de experimento.

- Baseline: como era antes da IA
- Hipótese e métrica de sucesso
- Dimensões: produtividade, qualidade, adoção, segurança, conhecimento e custo
- Por que métrica de uso isolada não prova valor
- Registro de experimento
- Decisão: escalar, ajustar ou descartar
- Rituais: revisão de prompts e demonstração de resultados
- Escalar prática validada, não ferramenta

PRÁTICA PROPOSTA

Escolher uma tarefa repetitiva, como escrever os testes de uma função. Fazer uma versão sem IA e outra com IA em funções equivalentes, para o aprendizado da primeira não distorcer a segunda. Medir tempo e defeitos e registrar a decisão no modelo de experimento.

ENTREGÁVEL

Registro de experimento preenchido com baseline, resultado e decisão.

3.7 Prática integrada ponta a ponta

Une os três eixos em uma entrega pequena, do item de backlog ao pull request. Refinar o item e os critérios de aceite vem da Agilidade, implementar em commits pequenos com testes e code review vem da carreira de Desenvolvedor, e o uso de IA com contexto, revisão adversarial e gates vem deste eixo. O ciclo fecha com retrospectiva e medição.

- Refinar o item e escrever critérios de aceite
- Montar o pacote de contexto para a IA
- Mapear padrões do projeto antes de codar
- Implementar em fragmentos com commits pequenos
- Testar e coletar evidências
- Autochecagem e revisão adversarial
- Abrir pull request com descrição objetiva
- Retrospectiva e registro do aprendizado

PRÁTICA PROPOSTA

Escolher um item pequeno de um backlog de treino, como validar um campo de formulário. Percorrer todas as etapas até um pull request simulado, registrando os prompts usados, a revisão feita e as evidências de teste.

ENTREGÁVEL

Pull request simulado com evidências, checklist de revisão preenchido e retrospectiva curta.

EIXO 03

Competências do eixo

Delegar com critério
Decidir o que a IA rascunha e o que exige decisão e responsabilidade humana.

Engenharia de contexto
Montar pedidos com contexto, restrições, critérios de aceite e formato de saída.

Revisão crítica
Validar código gerado quanto a padrão, segurança, tratamento de erro e testes.

Uso responsável
Proteger dados, segredos e licenças e respeitar a política da organização.

Orquestração
Combinar instruções, agentes, gates e evidências em um fluxo de entrega confiável.

Decisão por evidência
Medir o efeito da IA com baseline e decidir escalar, ajustar ou descartar.

EIXO 03 · INSIGHTS, ALERTAS E DICAS

Em destaque

● ALERTA

Antes de colar

Credenciais e tokens nunca vão para ferramentas de IA; se algum vazar, troque-o. Código proprietário, dados pessoais e informações de clientes só com contrato, configuração e aprovação da organização. Na dúvida, mascare ou pergunte.

● INSIGHT

Quem assina responde

A IA rascunha, a pessoa assina. Erro que passa para o code review não fica na conversa com o assistente: fica na confiança do time em quem entregou.

● DICA

Comece pequeno

Peça uma coisa por vez, em um fragmento que você consiga ler inteiro. Mudanças pequenas são mais fáceis de revisar, testar e desfazer.

● ALERTA

Confiança não é prova

Texto seguro não é texto correto. Confirme APIs, nomes e regras na documentação oficial ou executando o código antes de aceitar a resposta.

EIXO 03

Conceitos-chave

IA generativa	Tipo de inteligência artificial que gera texto, código ou imagens a partir de um pedido, prevendo o que é provável com base em padrões aprendidos no treinamento. Produz respostas plausíveis, não necessariamente corretas.
Copiloto	Modelo de uso em que a IA apoia a pessoa com sugestões e rascunhos, enquanto a pessoa decide, revisa e responde pelo resultado.
Prompt	Pedido enviado ao modelo. Um bom prompt traz contexto, papel, tarefa, restrições, critérios de aceite e formato esperado da resposta.

CONCEITOS-CHAVE (CONTINUAÇÃO)	
Engenharia de contexto	Prática de reunir e organizar as informações certas para o modelo: padrões do projeto, trechos de código, exemplos e restrições, de modo que a resposta seja útil e verificável.
Alucinação	Resposta que parece correta, mas inventa fato, função, API ou regra que não existe. Por isso toda saída precisa ser confirmada em fonte oficial ou por teste.
Janela de contexto	Quantidade de texto que o modelo considera de uma vez, incluindo o pedido e os arquivos enviados. O que fica fora da janela não influencia a resposta.
Agente de código	Programa baseado em modelo que lê arquivos, executa comandos e propõe alterações no projeto, seguindo instruções e dentro de limites definidos pela pessoa.
Orquestração	Coordenação de contexto, agentes, revisões e verificações para que o trabalho com IA siga um fluxo previsível, do pedido à entrega com evidências.
Instruções persistentes	Arquivo guardado no repositório do projeto, com tecnologias (stack), padrões, regras de teste e proibições, que o agente lê a cada sessão para manter o trabalho consistente.
Playbook	Roteiro reutilizável para um tipo de tarefa, como documentar um endpoint ou investigar um incidente, com passos, prompts e critérios de conclusão.
Verificação adversarial	Revisão em que alguém, pessoa ou agente, procura ativamente falhas, desvios de padrão e riscos no resultado, como faria um revisor exigente. Não substitui testes nem revisão humana.
Quality gate	Critério objetivo para avançar: build aprovado, testes passando e análise estática (leitura automática do código atrás de falhas e más práticas) sem pendências bloqueantes.

CONCEITOS-CHAVE (CONTINUAÇÃO)	
Baseline	Medida de referência de como o trabalho era feito antes da mudança. Sem baseline, não há como saber se a IA melhorou tempo ou qualidade.
Teste de caracterização	Teste que registra o comportamento atual de um código legado, certo ou errado, para detectar mudanças indesejadas durante a modernização.

EIXO 03 · VER ANEXOS

Modelos e checklists deste eixo

CHECKLIST 3-A	**Revisão de código gerado por IA**	Anexo B · p. 49
CHECKLIST 3-B	**Antes de usar IA**	Anexo B · p. 49
CHECKLIST 3-C	**Registro de experimento com IA**	Anexo B · p. 50
MODELO 3-D	**Prompt estruturado em seis blocos**	Anexo A · p. 44
MODELO 3-E	**Revisão técnica de código**	Anexo A · p. 45
MODELO 3-F	**Cenários de teste a partir de uma regra**	Anexo A · p. 45

Fluxo integrado

O fluxo abaixo acompanha um item de trabalho do começo ao fim e liga os três eixos: o Eixo 1 organiza o que fazer, o Eixo 2 constrói com qualidade e o Eixo 3 acelera com responsabilidade.

ETAPA	EIXO 01 Organizar	EIXO 02 Construir	EIXO 03 Acelerar
1 **Descobrir o problema** — Conversar com quem usa o produto, definir o problema e levantar hipóteses (suposições a testar) antes de investir em código.	E1		
2 **Organizar o backlog** — Transformar o que foi descoberto em itens pequenos, com critério de aceite (a condição para considerar o item pronto), ordenados por valor.	E1		
3 **Planejar a sprint** — Definir a meta da sprint, estimar em conjunto e puxar só o que o time consegue concluir. O quadro Kanban mostra o que está em andamento.	E1		
4 **Preparar o contexto** — Reunir critério de aceite, padrões do projeto e restrições. Esse contexto orienta a pessoa e o assistente de IA, que não conhece o projeto por conta própria.		E2	E3
5 **Construir em fragmentos** — Criar uma branch, um ramo isolado de trabalho, e implementar uma parte por vez, com código limpo e commits pequenos. O assistente de IA apoia e a pessoa decide.		E2	E3
6 **Provar e revisar** — Rodar os testes, abrir o pull request (pedido para incorporar a mudança) e passar por code review. O que a IA gerou segue o mesmo checklist.		E2	E3
7 **Entregar e aprender** — Apresentar o incremento (a parte pronta), registrar evidências, medir o efeito e levar as lições para a retrospectiva, reunião do time sobre o que melhorar, e para o próximo backlog.	E1	E2	E3

Matriz de competências

COMPETÊNCIA	DESCRIÇÃO	EIXO 01	EIXO 02	EIXO 03
Comunicar e colaborar	Alinhar expectativas, pedir ajuda com clareza, dar e receber feedback e registrar decisões para que o time inteiro acompanhe.	✓	✓	✓
Quebrar e priorizar o trabalho	Dividir uma necessidade em itens pequenos, com critério de aceite, e ordená-los pelo valor que entregam.	✓	·	·
Estimar e planejar em conjunto	Usar Scrum e Kanban para planejar a sprint, definir metas e enxergar o fluxo de trabalho.	✓	·	·
Versionar com Git	Trabalhar com branches, commits pequenos e pull requests, mantendo um histórico que outra pessoa consegue ler.	·	✓	·
Escrever código limpo	Usar nomes claros, funções pequenas e os padrões do projeto para que o código seja simples de entender e manter.	·	✓	·
Testar com critério	Escolher o que testar a partir do critério de aceite e escrever testes que protegem o comportamento esperado.	·	✓	·
Revisar código	Fazer e receber code review com comentários objetivos, incluindo a revisão de tudo o que foi gerado por IA.	·	✓	✓
Planejar a carreira	Entender os níveis de estágio e júnior, montar portfólio e definir um plano de estudo para o próximo passo.	·	✓	·
Dar contexto à IA	Preparar objetivo, restrições, padrões e critérios de aceite e escrever o prompt, ou seja, o pedido ao assistente, com clareza.	·	·	✓
Governar o uso de IA	Proteger dados e segredos, validar a saída do modelo e manter a decisão final com a pessoa responsável.	·	·	✓
Medir com evidência	Comparar antes e depois com métricas simples, como tempo, retrabalho e qualidade, para decidir o que manter.	✓	·	✓

Autoavaliação

EIXO 01 Agilidade e Gestão de Projetos

- [ ] Consigo explicar a diferença entre Scrum e Kanban e quando usar cada um.
- [ ] Consigo nomear os papéis, eventos e artefatos do Scrum e a função de cada um.
- [ ] Consigo transformar uma ideia em itens de backlog com critério de aceite.
- [ ] Consigo ordenar um backlog explicando o critério de prioridade.
- [ ] Consigo estimar um item em conjunto com o time e justificar a estimativa.
- [ ] Consigo conduzir os passos básicos de uma descoberta de produto (discovery): problema, hipótese e teste.
- [ ] Consigo registrar riscos com probabilidade, impacto, responsável e resposta escolhida.

EIXO 02 Atuar como Desenvolvedor: nível Estágio e Júnior

- [ ] Consigo criar uma branch, fazer commits pequenos no padrão Conventional Commits e abrir um pull request.
- [ ] Consigo mapear o padrão do projeto e conferir se a tarefa já está implementada antes de codar.
- [ ] Consigo resolver um conflito de merge simples.
- [ ] Consigo melhorar nomes e estrutura de um trecho de código sem mudar seu comportamento.
- [ ] Consigo escrever um teste automatizado para uma regra simples.
- [ ] Consigo revisar o pull request de outra pessoa com comentários objetivos e respeitosos.
- [ ] Consigo descrever meu nível atual e montar um plano de estudo para o próximo passo.

EIXO 03 Desenvolvimento e Orquestração de IA

- [ ] Consigo montar o contexto de uma tarefa antes de pedir ajuda ao assistente de IA.
- [ ] Consigo escrever um prompt com papel, tarefa, restrições e formato de saída.
- [ ] Consigo identificar quais dados nunca devem ser enviados a um assistente de IA.
- [ ] Consigo revisar o que o assistente gerou com um checklist: compila, segue o padrão e tem testes.
- [ ] Consigo dividir uma tarefa em fragmentos pequenos para conduzir um agente de código, que lê e altera arquivos do projeto.
- [ ] Consigo registrar o que foi pedido, aceito e validado, para ter rastreabilidade.
- [ ] Consigo medir o efeito do uso de IA comparando antes e depois.

Próximos passos

O aprendizado se consolida na prática repetida. Os horizontes abaixo são uma sugestão de ritmo para os próximos 90 dias. Ajuste ao seu tempo e ao seu objetivo.

HORIZONTE

30 dias

Fundamentos e hábito

- Criar um repositório próprio e usar Git todos os dias, com commits pequenos.
- Montar um quadro Kanban pessoal com backlog, em andamento e concluído.
- Ler o Guia Scrum e o Manifesto para Desenvolvimento Ágil de Software.
- Escolher três critérios da autoavaliação e treinar um por semana.

HORIZONTE

60 dias

Prática com entrega

- Construir um projeto pequeno em sprints curtas, cada uma com uma meta.
- Abrir pull requests pequenos e pedir code review a uma pessoa mais experiente.
- Escrever testes automatizados para as regras principais do projeto.
- Guardar os prompts que funcionaram, junto com o contexto usado.

HORIZONTE

90 dias

Portfólio e visibilidade

- Publicar o projeto com README, backlog e histórico de commits legível.
- Fazer uma retrospectiva pessoal: o que funcionou, o que mudar, o que estudar.
- Comparar o trabalho com e sem apoio de IA, usando tempo e retrabalho.
- Definir o plano de estudo e de candidaturas para o próximo ciclo.

GUIAS, LIVROS, SITES E PRÁTICA

Recursos para continuar

Guia Scrum (Scrum Guide) — GUIA
Guia oficial do Scrum, escrito por seus criadores. Curto, gratuito e referência para papéis, eventos e artefatos.

Manifesto para Desenvolvimento Ágil de Software — SITE
Quatro valores e doze princípios que deram origem à agilidade. Leitura de poucos minutos.

Kanban Guide — GUIA
Define Kanban e suas práticas: visualizar o fluxo, gerenciar o trabalho em andamento e melhorar o fluxo.

Pro Git — LIVRO
Livro gratuito e online sobre Git, dos fundamentos a branches e fluxos de trabalho.

Documentação oficial do Git — SITE
Referência de comandos e conceitos, para consultar quando um comando não se comporta como esperado.

Conventional Commits — SITE
Especificação curta para escrever mensagens de commit padronizadas, como feat e fix, que facilitam histórico e revisão.

Código Limpo (Clean Code) — LIVRO
Livro de Robert C. Martin sobre nomes, funções pequenas e legibilidade. Útil como vocabulário de code review.

OWASP Top 10 — GUIA
Lista pública dos riscos de segurança mais críticos em aplicações web. Serve também para revisar código gerado por IA.

Projeto pessoal com backlog e pull requests — PRÁTICA
Construir algo pequeno, com uma branch por tarefa e revisão do próprio pull request, transforma leitura em hábito.

Quem conduziu



MENTOR

Erick Prado

ARQUITETO DE SISTEMAS · ESPECIALISTA EM IA
GESTÃO DE PROJETOS · LIDERANÇA TÉCNICA

Erick Prado é arquiteto de sistemas e especialista em inteligência artificial, com atuação em gestão de projetos e liderança técnica. Acredita que a tecnologia só gera valor quando está integrada a uma gestão eficiente e a pessoas bem preparadas. Por isso ensina compartilhando método, exemplo e critério, para que quem está começando consiga se posicionar bem no mercado de tecnologia.

PRINCÍPIOS

Como ele ensina

01

Pessoas vêm antes de ferramentas.

02

Explicar o porquê antes do como.

03

Compartilhar método, e não só a resposta.

04

A IA apoia, e a responsabilidade continua humana.

Invista na tecnologia, desenvolva sua gestão, valorize sua equipe.

ERICK PRADO · NERDZONE SOLUTIONS

Sobre a NerdZone

A NerdZone é uma empresa de engenharia de software com alma nerd. Combina programação, visão de produto e IA para tirar ideias do papel e resolver problemas concretos de negócio. Seu ecossistema inclui o ZoneManager, plataforma web de back-office para operação e gestão de empresas, e o ZoneGo, aplicativo mobile do cliente final.

NERDZONE SOLUTIONS

Missão, visão e valores

MISSÃO

Colocar o conhecimento em movimento, com energia e curiosidade, para entregar soluções que fazem diferença no dia a dia das empresas.

VISÃO

Ser a equipe que faz diferença no negócio de quem atendemos, porque resultados grandes vêm de times e não de pessoas isoladas.

VALORES

Pessoas da equipe em primeiro lugar. O desenvolvimento de quem trabalha na NerdZone vem antes da publicidade, porque é o time que constrói a lealdade do cliente.

ANEXO A

Modelos

EIXO 01 Agilidade e Gestão de Projetos

MODELO 1-D

História de usuário com critérios de aceite

Padronizar a escrita de histórias para que negócio e time entendam o mesmo resultado.

```
Título: [verbo + objeto, curto]
Como [tipo de usuário],
quero [ação],
para [benefício].

Critérios de aceite
Cenário: [nome do cenário]
  Dado [contexto inicial]
  Quando [ação do usuário]
  Então [resultado observável]

Observações: [regras de negócio, dúvidas, link do mockup]
```

MODELO 1-E

Test card de hipótese

Registrar uma hipótese de discovery de forma que ela possa ser testada e decidida.

```
Hipótese: [resumo em uma frase]
Acreditamos que [usuário] [tem este problema ou necessidade].
Para verificar, vamos [experimento ou teste].
E vamos medir [métrica].
Estamos certos se [critério de sucesso].
Risco principal: [o que pode invalidar a hipótese]
Resultado: [validada | invalidada | inconclusiva]
Próxima decisão: [continuar, ajustar ou parar]
```

MODELO 1-F

Squad canvas em uma página

Reunir em um lugar o propósito, o domínio e os acordos de um time.

```
Propósito: [por que o time existe]
Domínio: [que parte do negócio e dos sistemas pertence ao time]
Responsabilidades-chave: [...]
Membros e papéis: [...]
Pessoas-chave do negócio: [...]
Ativos: [sistemas e produtos que o time cuida]
Metas: [resultados esperados no período]
Acordo de trabalho: [...]
Eventos e horários: [...]
Definition of Ready: [...]
Definition of Done: [...]
Identidade: [nome e símbolo do time]
```

EIXO 02 Atuar como Desenvolvedor: nível Estágio e Júnior

MODELO 2-D

Descrição de pull request

Dar ao revisor o que ele precisa para entender o diff e implantar a mudança, sem relatar o processo de desenvolvimento. Se o time tiver modelo próprio, use o do time.

```
Título: tipo(escopo): resumo curto no imperativo
## O que entrega
Uma ou duas frases sobre o resultado para o usuário ou para o sistema.
## O que muda
- Área, tela, classe ou endpoint: comportamento resultante.
## Regras e validações
- Condição, efeito e mensagem exibida.
## Banco e configuração
- Script, colunas ou chaves novas e o efeito. Escreva “nenhuma” se não houver.
## Para implantar
- Ordem, configuração e dependência de outro PR, citado pelo número e pelo assunto.
## Testes incluídos
- Arquivo ou classe de teste e o que cobre.
```

MODELO 2-E

Mensagem de pedido de ajuda

Permitir que quem ajuda entenda o problema em um minuto e responda sem precisar perguntar de volta.

```
Contexto: o que estou fazendo e em qual tarefa.
Objetivo: o resultado que eu esperava.
Resultado obtido: o que aconteceu, com a mensagem de erro completa.
Já tentei: 1 - ... | 2 - ...
Onde procurei: documentação, arquivos semelhantes e histórico.
Pergunta: uma pergunta objetiva, de preferência sim/não ou A/B.
```

MODELO 2-F

Resposta a comentário de review

Responder de forma objetiva e profissional, sem defensiva, mostrando o que foi feito e como foi validado.

```
Tipo: padrão | bug | preferência | dúvida de negócio
Alterado: o que mudou, em uma frase.
Motivo: por que isso alinha ao padrão do projeto ou corrige o problema.
Validação: comando executado e resultado.
Commit: referência do commit com a correção.
```

EIXO 03

Desenvolvimento e Orquestração de IA

MODELO 3-D

Prompt estruturado em seis blocos

Base para qualquer pedido de código, análise ou documentação.

```
Contexto: [sistema, domínio, stack e problema em 2 a 4 linhas]
Papel: Atue como [desenvolvedor sênior | arquiteto | QA | revisor].
Tarefa: [o que deve ser produzido, de forma específica e sem ambiguidade]
Restrições: [padrões do projeto, camadas, tecnologias permitidas, o que não alterar]
Critérios de aceite: [como saberei que a resposta está correta]
Formato: [código, tabela, checklist, plano de teste ou registro de decisão]
Antes de responder, liste suas dúvidas e suposições.
```

MODELO 3-E

Revisão técnica de código

Pedir uma revisão crítica antes do pull request, com achados priorizados.

```
Atue como revisor técnico sênior.
Analise o código abaixo considerando segurança, legibilidade, desempenho, aderência à arquitetura, testes e risco em produção.
Padrões do projeto: [camadas, nomenclatura, tratamento de erro]
Traga os achados ordenados por severidade: bloqueador, alto, médio e baixo.
Para cada achado informe local, problema, impacto e correção mínima.
Não proponha refatoração fora do escopo.
Se não houver bloqueadores, diga isso e liste o risco residual.
Código:
[cole o trecho, já sem dados sensíveis]
```

MODELO 3-F

Cenários de teste a partir de uma regra

Gerar casos positivos, negativos e de borda com dados fictícios.

```
Com base na regra de negócio e no código abaixo, gere cenários de teste positivos, negativos e de borda.
Regra: [descreva a regra em linguagem simples]
Separe em unitário, integração e regressão.
Para cada cenário informe entrada fictícia, ação e resultado esperado.
Use apenas dados fictícios, sem informação real de pessoas ou clientes.
Formato: tabela com cenário, tipo, entrada e resultado esperado.
Código:
[cole o trecho, já sem dados sensíveis]
```

ANEXO B

Checklists

EIXO 01 Agilidade e Gestão de Projetos

CHECKLIST 1-A

História pronta para entrar na sprint

Exemplo de Definition of Ready, prática do time que não faz parte do Guia Scrum. Usar no refinamento, antes de levar a história para a planning.

- [ ] Título claro e curto
- [ ] Descrição na voz do usuário: como, quero, para
- [ ] Critérios de aceite escritos em dado, quando, então
- [ ] Mockup ou exemplo anexado quando houver tela
- [ ] Dependências externas identificadas ou resolvidas
- [ ] Cabe em uma sprint e passa no INVEST
- [ ] Estimativa feita pelos Developers
- [ ] Dúvidas registradas e respondidas pelo Product Owner

CHECKLIST 1-B

Definition of Done do incremento

Exemplo genérico, a ajustar pelo time. Usar ao acordar o padrão de qualidade e ao revisar cada item antes de considerá-lo concluído.

- [ ] Critérios de aceite atendidos
- [ ] Código versionado e revisado por outra pessoa
- [ ] Testes automatizados executados e passando
- [ ] Bugs encontrados corrigidos e retestados
- [ ] Funcionando em ambiente de teste
- [ ] Validação de QA registrada
- [ ] Documentação técnica atualizada quando a mudança for relevante
- [ ] Aceite do solicitante, se o time acordou essa etapa

CHECKLIST 1-C

Preparação da sprint

Antes da planning, na véspera ou na manhã do evento.

- [ ] Product Goal visível e backlog ordenado pelo Product Owner
- [ ] Capacidade (capacity) do time calculada com ausências e reservas
- [ ] Histórias prontas em quantidade suficiente para a sprint
- [ ] Riscos e dependências mapeados
- [ ] Sprint Goal proposto e discutido
- [ ] Quadro atualizado e acessível a todos
- [ ] Definition of Done revisada

EIXO 02

Atuar como Desenvolvedor: nível Estágio e Júnior

CHECKLIST 2-A

Antes do pull request

Depois de terminar a mudança e antes de pedir revisão

- [ ] Li o diff inteiro contra a branch base, como se fosse de outra pessoa.
- [ ] A mudança resolve só o que a tarefa pede, sem refatoração fora do escopo.
- [ ] Segui o padrão de arquivos semelhantes do projeto: pastas, nomes, erros e contrato.
- [ ] Não há segredo, .env, dado pessoal, log de depuração nem código comentado no diff.
- [ ] Os nomes são claros; não sobrou código morto (trecho sem uso) nem comentário que repete o código.
- [ ] A validação de entrada está na camada certa e os erros seguem o padrão do projeto.
- [ ] Build, lint e testes do escopo foram executados e passaram; comando e resultado estão registrados.
- [ ] Há testes para o comportamento novo, com ao menos um caso de borda.
- [ ] Os commits são pequenos e têm mensagens no padrão do time.
- [ ] A descrição do pull request diz o que entrega, o que muda e o que é preciso para implantar.

CHECKLIST 2-B

Pedir ajuda bem

Antes de enviar uma dúvida a alguém do time

- [ ] Descrevi o objetivo da tarefa em uma frase.
- [ ] Reproduzi o problema e sei os passos exatos.
- [ ] Copiei a mensagem de erro completa, sem cortar.
- [ ] Listei o que já tentei e o que aconteceu em cada tentativa.
- [ ] Pesquisei na documentação, no código e no histórico do projeto.
- [ ] Removi segredos e dados pessoais do que vou enviar.
- [ ] Terminei com uma pergunta objetiva.

CHECKLIST 2-C

Responder a um comentário de review

Ao receber apontamentos em um pull request

- [ ] Entendi o comentário e o classifiquei: padrão, bug, preferência ou dúvida de negócio.
- [ ] Se faltou contexto, fiz uma pergunta fechada em vez de supor.
- [ ] Fiz a correção mínima suficiente, sem ampliar o escopo.
- [ ] Atualizei ou criei o teste afetado pela correção.
- [ ] Executei novamente build, lint e testes do escopo.
- [ ] Respondi dizendo o que mudou, por que e como validei.
- [ ] Se discordo, apresentei evidência e mantive o tom técnico.

EIXO 03

Desenvolvimento e Orquestração de IA

CHECKLIST 3-A

Revisão de código gerado por IA

Antes de commitar ou abrir pull request com trecho produzido por assistente de IA.

- [ ] Compila e executa localmente?
- [ ] Segue arquitetura, camadas e padrões do projeto?
- [ ] Não expõe segredo, dado sensível ou informação proprietária?
- [ ] Tem tratamento adequado de erro, log e observabilidade?
- [ ] Possui testes ou, no mínimo, uma estratégia de validação clara?
- [ ] Não cria dependência nova sem necessidade e sem aprovação?
- [ ] Cada dependência ou trecho sugerido existe de fato e tem origem e licença verificadas?
- [ ] Não altera comportamento existente sem teste de regressão?
- [ ] Você consegue explicar o que cada trecho faz e por quê?
- [ ] Foi revisado por pessoa responsável antes do merge?

CHECKLIST 3-B

Antes de usar IA

Antes de enviar qualquer conteúdo a uma ferramenta de IA.

- [ ] O conteúdo contém dado pessoal, segredo ou informação de cliente?
- [ ] A ferramenta tem contrato, política de retenção e opção de não treinar com seus dados?
- [ ] A política da organização permite esse caso de uso?
- [ ] Uma pessoa responsável vai revisar a saída?
- [ ] Existe teste ou evidência objetiva para confirmar o resultado?
- [ ] A decisão final fica com você e o time, e não com o modelo?
- [ ] O uso será registrado, quando a organização exigir?

CHECKLIST 3-C

Registro de experimento com IA

Ao testar uma ferramenta ou prática de IA em escopo pequeno, antes de decidir escalar.

- [ ] Nome do experimento
- [ ] Dor ou oportunidade que motivou o teste
- [ ] Hipótese a confirmar ou refutar
- [ ] Ferramenta ou modelo avaliado
- [ ] Dados utilizados e sua classificação
- [ ] Métrica de sucesso e baseline
- [ ] Resultado observado
- [ ] Decisão: escalar, ajustar ou descartar

ANEXO C

Glossário

Agente de código	Programa baseado em modelo que lê arquivos, executa comandos e propõe alterações no projeto, seguindo instruções e dentro de limites definidos pela pessoa.
Alucinação	Resposta que parece correta, mas inventa fato, função, API ou regra que não existe. Por isso toda saída precisa ser confirmada em fonte oficial ou por teste.
Backlog	Lista ordenada do que precisa ser feito. No Scrum, o Product Backlog é a lista do produto, cuidada pelo Product Owner, e o Sprint Backlog é o que o time escolhe fazer na sprint, com o plano para entregar.
Baseline	Medida de referência de como o trabalho era feito antes da mudança. Sem baseline, não há como saber se a IA melhorou tempo ou qualidade.
Branch	Linha paralela de desenvolvimento no Git. Permite alterar o código de uma tarefa sem afetar a linha principal até a mudança ser revisada e integrada à branch base, que a recebe.
Camadas, controller, DTO e repositório	Camadas dividem o código por responsabilidade. O controller recebe a requisição, o serviço aplica a regra de negócio, o repositório acessa o banco e o DTO é o objeto que carrega os dados que entram e saem da API.
Cobertura de testes	Percentual do código executado pelos testes. Indica o que foi exercitado, mas não prova que o comportamento está correto nem que os casos de borda foram cobertos.
Code review	Revisão do código por outra pessoa antes da integração. Busca erros, inconsistência com o padrão do projeto e riscos, e também compartilha conhecimento no time.
Code smell	Sinal no código de que o desenho pode estar ruim, como método longo, duplicação, nome confuso ou acoplamento alto. Nem sempre é um bug, mas dificulta a manutenção.
Commit	Registro de um conjunto de alterações no histórico do Git, com uma mensagem que explica o que mudou. Cada commit deve tratar de uma única preocupação.

GLOSSÁRIO (CONTINUAÇÃO)	
Complexidade cognitiva	Medida de quão difícil é entender o fluxo de uma função. Condicionais aninhadas, laços e tratamentos de exceção aumentam o valor e cansam quem lê.
Conventional Commits	Convenção para mensagens de commit no formato tipo(escopo): descrição, com tipos como feat, fix, docs, refactor, test e chore. Facilita leitura do histórico e automações.
Copiloto	Modelo de uso em que a IA apoia a pessoa com sugestões e rascunhos, enquanto a pessoa decide, revisa e responde pelo resultado.
Critério de aceite	Condição objetiva que um item precisa atender para ser considerado pronto. Pode ser escrito como cenário: dado, quando, então.
Definition of Done	Descrição formal do estado em que um incremento, a versão utilizável do produto, atende ao padrão de qualidade exigido. É o compromisso do incremento e vale para todos os itens.
Definition of Ready	Acordo do time sobre o que um item precisa ter para entrar em uma sprint. É uma prática comum, mas não faz parte do Guia Scrum.
Discovery	Conjunto de atividades para descobrir o que vale construir antes de investir em código: alinhar objetivos, pesquisar, idear, prototipar e refinar até o MVP, testando hipóteses com usuários.
Empirismo	Decidir com base no que se observa: formular hipótese, experimentar, inspecionar o resultado e adaptar. É a base do Scrum.
Engenharia de contexto	Prática de reunir e organizar as informações certas para o modelo: padrões do projeto, trechos de código, exemplos e restrições, de modo que a resposta seja útil e verificável.
Gate “já está implementado?”	Verificação feita antes de codar: buscar no código, nas branches e no histórico se o escopo da tarefa já existe. Evita retrabalho e pull request vazio.
História de usuário e BDD	História descreve uma necessidade na voz do usuário (como, quero, para). BDD escreve os critérios de aceite como cenários: dado, quando, então.

GLOSSÁRIO (CONTINUAÇÃO)	
IA generativa	Tipo de inteligência artificial que gera texto, código ou imagens a partir de um pedido, prevendo o que é provável com base em padrões aprendidos no treinamento. Produz respostas plausíveis, não necessariamente corretas.
Incremento	Passo concreto rumo ao Product Goal: a versão utilizável do produto que atende à Definition of Done. Pode haver mais de um por sprint.
Instruções persistentes	Arquivo guardado no repositório do projeto, com tecnologias (stack), padrões, regras de teste e proibições, que o agente lê a cada sessão para manter o trabalho consistente.
Janela de contexto	Quantidade de texto que o modelo considera de uma vez, incluindo o pedido e os arquivos enviados. O que fica fora da janela não influencia a resposta.
Kanban	Método para visualizar o fluxo de trabalho em um quadro, limitar o que está em andamento e melhorar a entrega continuamente. Pode ser usado sozinho ou junto com o Scrum.
Lead time e cycle time	Lead time vai do pedido à entrega. Cycle time vai do início do trabalho à conclusão. Ambos mostram previsibilidade e ajudam a achar gargalos.
Limite de WIP	WIP (work in progress) é a quantidade de itens em andamento. Limitá-lo reduz espera e troca de contexto e deixa os gargalos visíveis.
Linter	Ferramenta que analisa o código sem executá-lo e aponta erros prováveis e desvios de estilo, de acordo com regras configuradas no projeto.
Manifesto Ágil	Documento de 2001 com quatro valores e doze princípios para desenvolver software, que prioriza pessoas, software funcionando, colaboração com o cliente e resposta a mudanças.
MVP	Produto minimamente viável: a menor versão que permite testar uma hipótese com usuários reais e gerar aprendizado validado antes de investir mais.
Método STAR	Estrutura para contar uma experiência em entrevista: situação, tarefa, ação e resultado. Mantém a resposta objetiva e baseada em fatos.

GLOSSÁRIO (CONTINUAÇÃO)	
Orquestração	Coordenação de contexto, agentes, revisões e verificações para que o trabalho com IA siga um fluxo previsível, do pedido à entrega com evidências.
Playbook	Roteiro reutilizável para um tipo de tarefa, como documentar um endpoint ou investigar um incidente, com passos, prompts e critérios de conclusão.
Product Goal e Sprint Goal	Product Goal é o objetivo de longo prazo do produto e o compromisso do Product Backlog. Sprint Goal é o objetivo único da sprint, criado na planning, e o compromisso do Sprint Backlog.
Prompt	Pedido enviado ao modelo. Um bom prompt traz contexto, papel, tarefa, restrições, critérios de aceite e formato esperado da resposta.
Pull request	Pedido para integrar uma branch à base. Reúne o diff (as linhas alteradas), a descrição e a conversa de revisão, e só é concluído quando o time aprova.
Quality gate	Critério objetivo para avançar: build aprovado, testes passando e análise estática (leitura automática do código atrás de falhas e más práticas) sem pendências bloqueantes.
Scrum Team	Time único, sem sub-times, que entrega um incremento útil a cada sprint. Reúne um Product Owner (responsável pelo valor e pelo Product Backlog), um Scrum Master (responsável pela efetividade do Scrum) e os Developers, pessoas que criam o incremento: código, testes, design e análise.
Sprint	Ciclo de um mês ou menos, de duração constante, que contém todos os eventos do Scrum. Uma nova sprint começa logo após o fim da anterior.

GLOSSÁRIO (CONTINUAÇÃO)	
Teste de caracterização	Teste que registra o comportamento atual de um código legado, certo ou errado, para detectar mudanças indesejadas durante a modernização.
Teste unitário	Teste automatizado de uma pequena parte do código, como uma função, isolada do resto. Prepara os dados, executa a parte testada e verifica o resultado esperado.
Validação de entrada	Conferência de formato, obrigatoriedade e tamanho dos dados que chegam ao sistema, feita na camada de entrada. A regra de negócio fica em outra camada.
Variável de ambiente	Valor de configuração, como senha ou chave de acesso, definido fora do código. Costuma ser carregado de um arquivo .env que não vai para o Git; o repositório guarda só um exemplo sem valores reais.
Verificação adversarial	Revisão em que alguém, pessoa ou agente, procura ativamente falhas, desvios de padrão e riscos no resultado, como faria um revisor exigente. Não substitui testes nem revisão humana.

Compartilhando Recursos
INICIATIVA NERDZONE

NerdZone

Organizar, construir e acelerar são verbos de prática. Escolha um critério da autoavaliação, transforme em uma tarefa pequena e repita. A tecnologia conecta, a gestão organiza, mas são as pessoas que fazem acontecer.

Tecnologia com identidade

NERDZONE SOLUTIONS

SITE
nerdzonesolutions.com

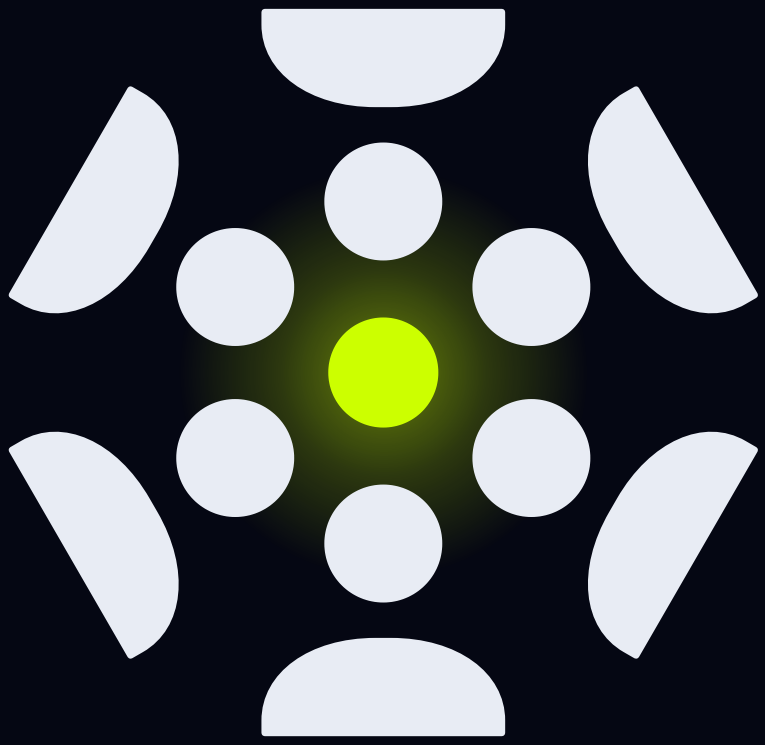
E-MAIL
contato@nerdzone.com.br

WHATSAPP
+55 11 93911-7113

LINKEDIN
linkedin.com/company/nerdzone-solutions

INSTAGRAM
@nerdzone.solutions

YOUTUBE
@NerdzoneIA



Conheça a iniciativa
Compartilhando Recursos
nerdzonesolutions.com/
compartilhando-recursos