

Para quem está começando em Salesforce

IA como *mentora técnica*

Como entender requisitos, investigar o ambiente, desenvolver com qualidade e evoluir usando IA sem terceirizar o próprio raciocínio.

PRINCÍPIO CENTRAL

A IA amplia sua capacidade de pensar, investigar, aprender e revisar. A leitura do requisito, o conhecimento do ambiente, a validação técnica e a responsabilidade pela entrega continuam sendo seus.

ANTES DE COMEÇAR

Chegou uma demanda nova. E agora?

Este material orienta devs júnior, pessoas em início de carreira em TI e quem está começando com Salesforce num cenário bem comum: chega uma especificação funcional, um chamado, uma user story, um requisito ou uma correção, e é preciso transformar um texto de negócio em uma mudança técnica segura.

A proposta vai além de "pedir código para a IA". Aqui você vai encontrar um processo de trabalho em que a IA funciona como **mentora, revisora, explicadora e parceira de investigação**. Quem desenvolve continua responsável por entender, questionar, testar, decidir e explicar a solução.

Esta versão também traz práticas observadas em demandas Salesforce e OmniStudio complexas: separar requisito literal de hipótese, investigar o cenário AS-IS antes de desenhar o TO-BE, rastrear dependências ponta a ponta, transformar dúvidas em verificações objetivas e alimentar a IA com evidências reais à medida que a investigação avança.

REGRA DE CONCLUSÃO

Se você não consegue explicar o que a solução faz, por que ela foi escolhida, o que ela impacta e como foi validada, a demanda ainda não terminou, mesmo que o código aparentemente funcione.

O CAMINHO

Siga o fluxo do capítulo 03 sempre que uma nova demanda chegar. Os outros capítulos aprofundam investigação, prompts, qualidade, segurança e evolução profissional.

- | | |
|---|---|
| 01 O papel correto da IA | 08 Validar código, configuração e arquitetura |
| 02 Antes de abrir a IA | 09 Testes, revisão e Definition of Done |
| 03 Da demanda à entrega | 10 Segurança, dados e limites |
| 3A Roteiro de análise de uma demanda | 11 Quando pedir ajuda humana |
| 04 Investigar um ambiente Salesforce | 12 Exemplo completo de uma demanda |
| 05 Conversar com a IA sem deixá-la inventar | 13 Checklist de bolso |
| 06 Prompts para o dia a dia | 14 Trilha de evolução do júnior |
| 07 Aprender enquanto desenvolve | 15 Fontes oficiais |

01

CAPÍTULO 01

O papel correto da IA no desenvolvimento

A IA acelera muito o entendimento: explica conceitos, sugere caminhos de investigação, gera consultas, revisa hipóteses, identifica riscos, propõe testes e discute alternativas. O problema começa quando ela vira o "cérebro terceirizado" do desenvolvedor.

USO SAUDÁVEL DA IA	USO QUE IMPEDE EVOLUÇÃO
Explicar um conceito e depois pedir que você o explique de volta.	Gerar uma solução inteira e copiar sem entender.
Ajudar a formular perguntas para investigar a org.	Aceitar nomes de objetos, campos, classes ou processos que a IA inventou.
Revisar uma hipótese criada por você.	Perguntar "qual é a solução?" antes de entender o problema.
Gerar uma SOQL ou comando e explicar cada parte.	Executar scripts com DML porque "a IA disse que está certo".
Comparar alternativas e mostrar riscos.	Usar a resposta mais convincente como prova.
Revisar seu código e apontar problemas.	Pedir código sem fornecer contexto do ambiente.
Ajudar a criar casos de teste e cenários negativos.	Considerar "compilou" como sinônimo de "está correto".

POSTURA OBRIGATÓRIA

Duvidar da IA é tratá-la como qualquer outra fonte técnica: perguntar de onde veio a afirmação, confrontar com o ambiente real, consultar a documentação oficial e testar antes de concluir. A ferramenta continua na mesa, só que sob a sua supervisão.

Quatro papéis que a IA pode exercer

i

MENTORA

Explica conceitos, faz perguntas e ajuda você a construir raciocínio.

ii

INVESTIGADORA ASSISTIDA

Ajuda a desenhar queries, buscas, comandos e caminhos de diagnóstico.

iii

REVISORA

Critica solução, código, arquitetura, testes e documentação.

iv

ACELERADORA

Automatiza tarefas mecânicas depois que o problema já está entendido.

02

CAPÍTULO 02

O que fazer antes de abrir a IA

A primeira leitura da demanda, seja ela um chamado, uma user story ou uma especificação funcional, deve ser humana. Quem entrega o texto para a IA antes de formar qualquer entendimento próprio perde justamente a etapa que mais desenvolve capacidade analítica.

Leitura manual em três passadas

- 01 **Primeira passada.** Leia do início ao fim sem tentar resolver. O objetivo é descobrir qual problema de negócio está sendo descrito.
- 02 **Segunda passada.** Marque atores, entradas, saídas, regras, exceções, telas, integrações, dados, critérios de aceite e palavras que você não entende.
- 03 **Terceira passada.** Escreva com suas palavras o que você acredita que precisa mudar e liste tudo que ainda não sabe.

ANTES DA IA, PRODUZA ESTAS CINCO LINHAS

1. O problema é...
2. Hoje parece funcionar assim...
3. A mudança solicitada parece ser...
4. Eu ainda não entendi...
5. Para confirmar, preciso olhar...

Perguntas manuais mínimas

- Qual é o comportamento atual?
- Qual é o comportamento esperado?
- Quem executa essa jornada e em qual momento?
- Qual dado entra e qual resultado deve sair?
- Existe critério de aceite explícito?
- Há exceções, regras condicionais ou cenários alternativos?
- A demanda fala em componente novo ou alteração de algo existente?
- Quais nomes de tela, objeto, campo, produto, integração ou processo aparecem?
- O que é requisito confirmado e o que estou apenas imaginando?
- Que perguntas eu faria para um analista funcional, tech lead ou usuário de negócio?

Classifique a dúvida antes de tentar respondê-la

Nem toda dúvida deve ir para a IA. Classificar evita pedir ao modelo uma resposta que só o negócio ou o próprio ambiente pode fornecer.

TIPO	EXEMPLO	FONTE CORRETA PARA RESPONDER
Negócio	"A regra também vale para contrato cancelado?"	Analista funcional, PO, usuário de negócio ou responsável pelo requisito.
Técnica	"Como um Integration Procedure trata este input?"	Documentação oficial, código e configuração existentes, testes e IA como apoio.
Ambiente	"Já existe algum Flow ou Apex atualizando este campo?"	Org, metadata recuperada, repositório, logs e buscas no projeto.

REGRA SIMPLES

Dúvida de negócio não deve ser "resolvida" pela IA. Dúvida de ambiente não deve ser respondida por memória ou suposição. Dúvida técnica pode usar IA, mas precisa ser confrontada com documentação oficial e com o padrão real do projeto.

03

CAPÍTULO 03

Fluxo completo: da demanda à entrega

Este fluxo deve virar hábito. Ele serve para correção de bug, melhoria, nova feature, integração, automação, LWC, Apex, OmniStudio, Flow ou configuração.

- 01 **Ler.** Entender manualmente o problema e registrar dúvidas.
- 02 **Estruturar.** Separar fatos, hipóteses, lacunas e critérios de aceite.
- 03 **Contextualizar a IA.** Entregar à IA somente o que já foi observado e exigir que ela não invente.
- 04 **Investigar.** Mapear o que existe na org e coletar evidências.
- 05 **Analisar impacto.** Descobrir dependências, consumidores, dados, permissões, automações e integrações.
- 06 **Desenhar.** Comparar opções e decidir a menor mudança segura.
- 07 **Implementar em partes.** Construir pequenos incrementos, entendendo cada alteração.
- 08 **Testar.** Validar cenário feliz, negativos, regressão e segurança.
- 09 **Revisar.** Usar IA, Code Analyzer e revisão humana quando aplicável.
- 10 **Explicar e registrar.** Conseguir contar a solução e guardar o aprendizado.

O quadro que evita alucinação

Durante toda a demanda, mantenha uma tabela simples. Ela força você e a IA a reconhecerem o nível de certeza de cada informação.

CATEGORIA	O QUE SIGNIFICA	EXEMPLO	AÇÃO
FATO	Está explícito na demanda ou foi confirmado por pessoa responsável.	"O campo deve ser obrigatório quando X."	Implementar ou validar diretamente.
EVIDÊNCIA	Foi observado no ambiente, metadata, log, query ou teste.	"O OmniScript chama o IP Y."	Guardar como prova do comportamento atual.
HIPÓTESE	Explicação possível, ainda não confirmada.	"Talvez o valor seja calculado no IP."	Investigar antes de decidir.
DÚVIDA	Informação necessária que ainda falta.	"Essa regra também vale para renovação?"	Perguntar ou pesquisar.

CATEGORIA	O QUE SIGNIFICA	EXEMPLO	AÇÃO
DECISÃO	Escolha técnica tomada após evidências.	"A alteração ficará no IP para não duplicar regra."	Registrar motivo e impacto.

Triagem de pendências: nem tudo vira pergunta

Uma prática madura é revisar a lista de dúvidas continuamente e retirar tudo o que já foi respondido pelo requisito, pelo source ou pela investigação. O objetivo é chegar apenas ao que realmente continua sem resposta.

STATUS	O QUE SIGNIFICA	PRÓXIMA AÇÃO
Respondido pelo requisito	A demanda já define a regra de forma literal.	Não perguntar de novo. Transformar em critério de aceite ou teste.
Respondido pelo source	Código, metadata, log ou configuração já provou o comportamento atual.	Registrar a evidência e seguir.
Buscar na org	A resposta existe tecnicamente, mas ainda não foi localizada.	Criar uma verificação objetiva: busca, query, retrieve, log ou inspeção.
Dúvida funcional	A resposta exige decisão ou esclarecimento de negócio.	Levar uma pergunta objetiva para a pessoa responsável.
Decisão técnica	O requisito diz o que precisa acontecer, mas não determina a implementação.	Comparar alternativas, padrões existentes, riscos e impacto antes de escolher.

UMA PERGUNTA BOA ELIMINA INCERTEZA

Antes de encaminhar uma dúvida, pergunte: isso já está respondido em algum lugar? É uma regra de negócio, uma informação do ambiente ou uma decisão técnica? Qual evidência encerraria esta dúvida?

Caderno de evidências da demanda

Em demandas grandes, mantenha um registro curto do que foi confirmado. Isso evita voltar para hipóteses antigas, ajuda a IA a acompanhar a investigação e facilita code review, QA e handoff.

PERGUNTA	EVIDÊNCIA COLETADA	CONCLUSÃO PERMITIDA	AINDA NÃO PROVA
Quem chama este cálculo?	Busca no projeto + metadata recuperada.	Existe uma referência direta no componente X.	Que seja o único caminho em runtime.
Onde o valor é persistido?	Payload antes/depois + registro consultado.	O campo Y recebe o valor após a etapa Z.	O comportamento em reprocessamento.

3A

CAPÍTULO 3A

Roteiro operacional de análise de uma nova demanda

Ler, formar entendimento próprio, confrontar o documento, transformar dúvidas em perguntas verificáveis, investigar o ambiente, voltar às pendências com evidências, refinar a redação e só então fechar o escopo técnico. O ponto central é trabalhar em rodadas, sem tentar resolver tudo de uma vez.

REGRA PRINCIPAL

A primeira lista de dúvidas é um rascunho, nunca a lista final. Ela precisa ser depurada à medida que novas evidências aparecem.

RODADA 0

Crie o caderno da demanda

Antes da primeira conversa com a IA, abra um espaço único para acompanhar a análise: planilha, documento ou página de notas. O objetivo é evitar que fatos, hipóteses e decisões se misturem ao longo dos dias.

- Mantenha o texto original da demanda preservado. Não reescreva o requisito e depois trate sua própria interpretação como se fosse o original.
- Crie uma área chamada "**Meu entendimento**" e escreva o que você acredita que será alterado.
- Crie uma área de "**Dúvidas minhas**". Escreva em primeira pessoa aquilo que realmente não ficou claro para você.
- Crie uma área de **evidências**: query, metadata, trecho de código, payload, log, print ou teste que respondeu alguma pergunta.
- Crie uma área de **decisões**: funcional, técnica e arquitetural. Registre quem decidiu e por quê quando isso for relevante.

Por que funciona: ao separar o documento original da sua leitura, você consegue voltar e perguntar "isso está realmente escrito ou fui eu que concluí?".

RODADA 1

Leia sozinho e escreva o que acha que terá de fazer

Depois da leitura geral, transforme o requisito em uma explicação sua. Ainda não procure solução. O objetivo é testar se você entendeu o problema.

- Escreva o comportamento atual que você acredita existir.
- Escreva o comportamento esperado descrito pela demanda.
- Liste o que parece novo, o que parece alteração de comportamento existente e o que aparentemente deve permanecer igual.

- Liste os componentes, objetos, campos, telas e regras citados literalmente na demanda.
- Marque com **HIPÓTESE** qualquer componente que você apenas imagina que participe.
- Liste suas dúvidas antes de perguntar qualquer coisa à IA.

Um bom teste é conseguir responder em voz alta: **"o problema hoje é X; depois da demanda esperamos Y; ainda não sei Z"**.

RODADA 2

Use a IA para confrontar sua leitura

A primeira utilidade da IA é apontar lacunas no seu entendimento. Passe a demanda, seu resumo e suas dúvidas. Peça comparação, não implementação.

PROMPT Confrontar minha leitura da demanda

Vou te passar a demanda, o meu entendimento e minhas dúvidas.
Não proponha solução ainda. Compare meu entendimento com o texto e devolva:
(1) o que está literalmente definido;
(2) o que eu interpretei corretamente;
(3) o que eu estou inferindo sem evidência;
(4) o que está ambíguo ou contraditório;
(5) o que precisa ser investigado no ambiente;
(6) o que precisa de decisão funcional.
Para cada conclusão, diga em que evidência você está se apoiando.

RODADA 3

Higienize as pendências antes de investigar

Um dos hábitos mais importantes. Antes de manter qualquer item na lista de pendências, tente destruí-lo. Uma pendência só sobrevive se você realmente não encontrou resposta.

- Procure primeiro na própria demanda. Muitas perguntas surgem porque um detalhe está respondido em outro RF, critério de aceite, tabela, anexo ou exemplo.
- Procure no histórico da análise. Não mantenha uma pergunta que já foi respondida numa rodada anterior.
- Veja se duas pendências são, na prática, a mesma dúvida escrita de formas diferentes. Se forem, una ou remova a duplicada.
- Se uma pendência mistura dois assuntos, divida. Pergunta composta costuma gerar resposta incompleta.
- Se o item não exige resposta funcional, mas representa algo que precisará ser tratado tecnicamente, reclassifique como impacto técnico.
- Se a demanda já tomou a decisão, mas o texto está mal escrito ou incompleto, registre como "decisão definida, precisa ser explicitada no requisito", e não como decisão em aberto.

Exemplos de refinamento: separar "precisão do percentual de desconto" de "limite técnico de casas decimais do campo"; separar "quantidade informada pelo usuário" de "quantidade usada no cálculo"; tratar a regra de itens já em promoção como subitem da regra de desconto; remover

uma pendência duplicada quando a preocupação já estava coberta por outra pergunta e manter apenas o impacto técnico associado.

Faça perguntas atômicas

Uma boa pendência permite uma resposta objetiva. Se a pergunta tiver dois ou três "e", provavelmente precisa ser quebrada.

PERGUNTA	
RUIM	"Como fica o desconto, a faixa de quantidade e o total quando o vendedor altera a cotação?"
MELHOR 1	"Qual percentual de desconto precisa ser gravado em cada item da cotação?"
MELHOR 2	"Como a faixa de desconto é definida a partir da quantidade?"
MELHOR 3	"Qual regra se aplica quando o item já tem um preço promocional?"

RODADA 4

Transforme cada dúvida de ambiente em investigação executável

"Precisa validar na org" não é resposta suficiente. Toda validação deve virar um procedimento objetivo.

- › O que exatamente eu preciso descobrir?
- › Onde eu vou procurar primeiro?
- › Qual query, busca, metadata, log, payload ou teste pode responder?
- › Qual resultado confirmaria a hipótese A?
- › Qual resultado descartaria a hipótese A?
- › O que esta evidência ainda não provaria, mesmo se eu encontrá-la?

EXEMPLO

Em vez de "validar se o campo participa do cálculo", escreva: buscar o API Name no projeto; recuperar a metadata atual; identificar referências em Flow, Apex e OmniStudio; executar um cenário controlado e observar o payload ou log para confirmar o uso em runtime.

Quando a IA disser "precisa validar", responda com quatro perguntas

- 01 Validar exatamente o quê?
- 02 Onde eu consigo validar isso no Salesforce ou no projeto?
- 03 Qual evidência devo trazer de volta?

04 Que resultado fecha a dúvida e que resultado abre uma nova investigação?

Assim "precisa validar" deixa de ser um estacionamento de perguntas vagas.

RODADA 5

Reconstrua o AS-IS e siga o dado ponta a ponta

Antes de projetar o TO-BE, prove como a jornada funciona hoje. Em demandas de cálculo, preço, desconto, quantidade, datas ou configuração, siga o dado como se estivesse rastreando um pacote sendo transportado pelo sistema.

- › Onde o usuário informa ou altera o dado?
- › Em qual estrutura ele aparece primeiro: atributo, JSON, campo, variável, linha de tabela?
- › Qual componente recebe esse valor depois?
- › Há transformação, agregação, cálculo ou conversão?
- › Onde o valor é persistido?
- › Quem o lê depois: rotina de cálculo, Flow, Apex, Integration Procedure, Data Mapper, LWC, integração?
- › O que acontece em edição, recálculo, reabertura, cópia, renovação ou reprocessamento?
- › Existe caminho legado que deve continuar funcionando quando a nova condição não se aplica?

IMPORTANTE

Não invente o próximo elo. Se você sabe que um OmniScript chama algo, descubra o nome real da chamada. Se sabe que um IP grava dados, descubra qual elemento e qual objeto. Cada elo precisa de evidência.

RODADA 6

Volte às pendências com evidências e comentários seus

Depois do discovery técnico, não siga direto para o código. Faça uma segunda rodada de análise: releia cada pendência, compare com o que descobriu e escreva sua dúvida atualizada com suas próprias palavras.

- Se a evidência respondeu, feche o item e registre a evidência.
- Se a evidência respondeu apenas parte, reescreva a pergunta para deixar só o que falta.
- Se surgiu contradição entre requisito e ambiente, registre a divergência explicitamente.
- Se você entendeu a pergunta de forma errada, corrija o texto. Não preserve uma pendência ruim só porque ela já estava numerada.
- Se um item virou impacto técnico, mova-o de categoria.
- Se o item depende de decisão humana, formule a pergunta para que analista ou PO consiga responder objetivamente.

Esse momento é especialmente valioso para o aprendizado: primeiro você vê a leitura da IA, depois vê o ambiente, e então escreve sua própria interpretação. O júnior deixa de apenas consumir respostas e passa a construir uma visão técnica.

A planilha de pendências deve ser viva

A planilha funciona como o controle de raciocínio da demanda, muito além de uma lista de dúvidas. Atualize a cada rodada. Uma estrutura útil:

COLUNA	O QUE REGISTRAR
ID	Identificador estável enquanto o item existir.
Tema	Assunto curto, por exemplo "desconto", "persistência", "cálculo", "UI".
Trecho da demanda	Texto que originou a análise.
Meu entendimento	Como você interpretou aquele trecho.
Dúvida minha	Pergunta escrita por você, em linguagem direta.
Classificação	Requisito / ambiente / funcional / decisão técnica / impacto técnico.
Evidência atual	Query, código, metadata, payload, log, teste ou resposta funcional.
O que a evidência prova	Conclusão permitida.
O que não prova	Limite da conclusão.
Próxima ação	Exatamente o próximo passo.
Status	Aberto, investigando, respondido pelo requisito, respondido por evidência, aguardando funcional, aguardando decisão técnica, duplicado/removido.
Impacto	Onde essa definição interfere na implementação ou nos testes.

REGRA

Nunca deixe uma célula com apenas "validar". Escreva validar o quê, onde e com qual evidência.

RODADA 7

Feche e reclassifique cada pendência

Antes de dizer que a análise terminou, toda pergunta precisa estar em um estado que faça sentido:

STATUS	QUANDO USAR
FECHADO PELO REQUISITO	A resposta já estava no requisito.

STATUS	QUANDO USAR
FECHADO POR EVIDÊNCIA	A org, código, metadata, log, payload ou teste respondeu.
AGUARDANDO FUNCIONAL	Depende de regra de negócio não definida.
AGUARDANDO DECISÃO TÉCNICA	Existem alternativas válidas e o time precisa escolher.
INVESTIGAÇÃO ABERTA	Ainda existe uma ação concreta de discovery.
IMPACTO TÉCNICO CONFIRMADO	Não é pergunta: é algo que a solução precisa considerar.
DUPLICADO / REMOVIDO	O item foi absorvido por outro ou não faz mais sentido.

Isso evita uma lista enorme de "pendências" em que metade já está resolvida e ninguém sabe mais o que realmente bloqueia o desenvolvimento.

RODADA 8

Gere o brief de implementação antes de codificar

Quando as pendências críticas estiverem fechadas ou conscientemente aceitas, escreva uma página de implementação. Se você não consegue produzir esse resumo, ainda não entendeu a demanda o suficiente.

- ☐ Objetivo da mudança em linguagem simples.
- ☐ AS-IS confirmado.
- ☐ TO-BE confirmado pela demanda e pelas decisões tomadas.
- ☐ Componentes que serão alterados.
- ☐ Componentes que podem ser impactados sem alteração direta.
- ☐ Dados que entram, transformações e onde persistem.
- ☐ Regras, exceções e comportamento legado que deve permanecer.
- ☐ Pendências restantes e se bloqueiam ou não o desenvolvimento.
- ☐ Estratégia de implementação em pequenas etapas observáveis.
- ☐ Cenários de teste funcional, regressão, permissão, recálculo/reentrada e limites relevantes.

O ritual para repetir em toda nova demanda

- 01 Leia sozinho e escreva o que entendeu.
- 02 Liste dúvidas sem pedir solução.
- 03 Use a IA para confrontar sua leitura.
- 04 Elimine perguntas já respondidas e duplicadas.
- 05 Quebre perguntas compostas.

- 06 Classifique o que é requisito, ambiente, funcional, decisão técnica ou impacto.
- 07 Transforme cada dúvida de ambiente em uma ação verificável.
- 08 Reconstrua o AS-IS e rastreie o dado ponta a ponta.
- 09 Traga as evidências para a IA e atualize as hipóteses.
- 10 Reescreva as pendências com seus próprios comentários.
- 11 Feche ou reclassifique cada item.
- 12 Gere o brief de implementação.
- 13 Desenvolva em passos pequenos, entendendo cada mudança.
- 14 Teste requisito, regressão e caminhos alternativos.
- 15 Explique a solução sem consultar a IA.

SINAL DE MATURIDADE

Usar a IA para acelerar investigação e revisão sem terceirizar o julgamento técnico.

04

CAPÍTULO 04

Como investigar um ambiente Salesforce

A IA não conhece a sua org. Mesmo conhecendo Salesforce, ela não sabe quais customizações, nomes de metadata, regras antigas, integrações, permissões, pacotes ou decisões arquiteturais existem na empresa. Por isso, a investigação precisa produzir contexto real.

Comece pela jornada, não pelo código

- Onde o usuário inicia a ação?
- Qual tela, OmniScript, Flow, LWC, Quick Action, FlexCard ou página participa?
- Qual componente recebe o primeiro input?
- Que chamadas acontecem depois?
- Quais objetos e campos são lidos ou gravados?
- Existe integração externa?
- Onde ficam as regras de negócio?
- Quem consome o resultado?
- Há perfis, permission sets, sharing ou FLS envolvidos?
- Há automações paralelas que também atualizam o mesmo registro?

Primeiro AS-IS, depois TO-BE

Antes de discutir como desenvolver, reconstrua como a jornada funciona hoje. Em projetos complexos, o requisito costuma mencionar apenas um pedaço visível, enquanto o comportamento real atravessa várias camadas. Se o AS-IS estiver errado, a solução será desenhada sobre uma premissa falsa.

PERGUNTA-CHAVE

Em vez de perguntar à IA "onde essa regra está na minha org?", pergunte "como eu provo onde essa regra está?". A resposta deve virar busca, query, inspeção de metadata, log, payload, teste ou pergunta funcional.

Rastreie o fluxo ponta a ponta

Um bom mapa técnico segue o dado e a execução. Nem toda demanda terá todas as camadas abaixo, mas a investigação deve continuar até você saber onde o valor nasce, onde muda, onde é usado e onde persiste.

01 Ação do usuário Botão, tela, edição, cálculo, save, reprocessamento.

02 Interface LWC, OmniScript, FlexCard, Screen Flow, Lightning page.

- 03 Orquestração** Integration Procedure, Flow, Remote Action, action chain.
- 04 Regra / cálculo** Apex, Expression Set, Decision Matrix, fórmulas, transform de Data Mapper.
- 05 Dados** Objetos, campos, relacionamentos e registros envolvidos (por exemplo Opportunity, Quote, Order).
- 06 Persistência** Onde o resultado final ou a configuração fica gravado.
- 07 Consumidores** Quem usa esse valor depois: UI, cálculo seguinte, documento, integração, aprovação.
- 08 Reprocessamento** O que acontece ao editar, recalcular, reabrir, duplicar ou usar registro antigo.
- 09 Assíncrono / histórico** Jobs, batch/queueable, snapshots, versões, dados históricos e timing.

Mapa de investigação por camada

CAMADA	O QUE PROCURAR	EVIDÊNCIAS ÚTEIS
Interface	Lightning page, LWC, FlexCard, OmniScript, Screen Flow.	Nome técnico, print, XML/JSON, inputs e outputs.
Orquestração	Flow, Integration Procedure, actions de OmniScript, Apex.	Ordem de execução, condições, payloads, logs.
Dados	Objects, fields, relationships, record types, metadata/configuração.	SOQL, Object Manager, schema, amostra de dados anonimizada.
Regra de negócio	Fórmulas, Apex, Decision Matrix, Calculation Procedure, Validation Rules, Flows.	Trechos de metadata, critérios e cenários.
Integração	Named Credentials, APIs, events, callouts, middleware.	Contrato, request/response, status, logs, tratamento de erro.
Segurança	Permission Sets, Profiles, FLS, sharing, running user.	Permissões efetivas e contexto de execução.
Dependências	Componentes que chamam ou são chamados.	Busca no projeto, Where is this used, Tooling/Metadata APIs, referências.
Qualidade	Testes, validações, análise estática, logs.	Apex tests, Jest, Code Analyzer, debug logs, evidência de QA.

Ferramentas práticas de investigação

- **Busca no projeto:** use "Find in Files" pelo API Name de campo, objeto, classe, LWC, Integration Procedure ou OmniScript.
- **Object Manager e Setup:** confirme tipo de campo, relacionamentos, fórmulas, regras, permissões e recursos disponíveis.
- **SOQL:** investigue dados reais e relações. A IA pode ajudar a montar a consulta, mas você revisa objeto, campos, filtros e volume.
- **Tooling API e metadata:** úteis para investigar classes, triggers e componentes quando a pergunta é sobre configuração e código, e não sobre registros de negócio.
- **Salesforce CLI:** consulta dados, recupera metadata e permite análise local do projeto.
- **Logs e payloads:** use evidência de execução para entender o caminho real, especialmente em OmniStudio e integrações.
- **Salesforce Code Analyzer:** análise estática como uma das camadas de revisão, sem substituir os testes.

CUIDADO IMPORTANTE

"Não encontrei" não significa "não existe". Pode estar fora do projeto local, em pacote gerenciado, em metadata ainda não recuperada, em integração externa, em configuração dinâmica ou em recurso ao qual você não tem acesso. Registre a limitação.

O ciclo de investigação assistida

- 01 **Formule uma pergunta verificável:** "de onde vem este valor?", "quem atualiza este campo?", "o que chama esta classe?".
- 02 **Peça à IA o caminho de investigação, não a conclusão:** busca, query, comando, log, payload ou metadata a inspecionar.
- 03 **Execute você mesmo** no ambiente correto e leia o resultado.
- 04 **Volte para a IA com a evidência real,** incluindo o que o resultado prova e o que ainda não prova.
- 05 **Atualize o mapa do fluxo** e elimine hipóteses que deixaram de fazer sentido.
- 06 **Só então** formule a próxima pergunta.

INVESTIGAÇÃO É ITERATIVA

O contexto da conversa deve crescer com fatos novos. A IA não "aprende a org" magicamente: você a torna útil ao devolver resultados de queries, trechos reais de metadata, payloads, logs e decisões funcionais confirmadas.

Quando documentação, código local e runtime não batem

Trate divergência como sinal de investigação, nunca como licença para escolher a versão que parece mais conveniente. Confirme branch/commit, versão ativa, metadata atual da org e evidência de execução. Um arquivo antigo no repositório pode não representar o runtime; uma tela pode exibir uma versão ativa diferente da documentação; um requisito pode descrever o desejado e não o que existe hoje.

- Registre qual fonte você consultou e, quando relevante, a versão/data.
- Recupere a metadata atual quando houver dúvida sobre o que está realmente implantado.
- Use log, payload ou teste para confirmar comportamento em runtime quando a configuração sozinha não basta.
- Não "corrija" a divergência por inferência. Primeiro prove qual estado é real e depois decida com o time.

Exemplos de comandos úteis

TERMINAL Consulta de dados com Salesforce CLI

```
sf data query --query "SELECT Id, Name FROM Account LIMIT 10" --target-org [minha-org]
```

O comando é apenas um exemplo. Ajuste objeto, campos, filtros e org. Para objetos da Tooling API, a CLI possui a opção `--use-tooling-api`.

TERMINAL Recuperação de metadata específica

```
sf project retrieve start --metadata ApexClass:[MinhaClasse] --target-org [minha-org]
```

Recupere somente o necessário quando possível e use controle de versão para comparar alterações.

05

CAPÍTULO 05

Como conversar com a IA sem deixá-la inventar

Um bom prompt é uma forma de fornecer contexto e impor um método de trabalho, sem frase mágica. Para desenvolvimento, ele precisa deixar claro o papel da IA, o que já se sabe, o que ainda é desconhecido, qual tarefa será feita e quais limites não podem ser ultrapassados.

Regra de resposta da IA

Peça para a IA classificar o que diz. Este formato funciona muito bem:

ETIQUETA	SIGNIFICADO
FATO DO CONTEXTO	Somente aquilo que você forneceu ou mostrou.
CONHECIMENTO GERAL	Comportamento documentado da plataforma, de preferência com fonte oficial.
HIPÓTESE	Possibilidade que depende de validação na org.
PRECISA VALIDAR	Algo que a IA não consegue confirmar com o material disponível.
PERGUNTA	Informação que você precisa buscar ou perguntar antes de avançar.

PROMPT BASE Transformar a IA em mentora técnica

Atue como meu mentor técnico Salesforce. Eu sou desenvolvedor júnior e quero aprender enquanto resolvo a demanda.

Regras obrigatórias:

1. Não invente objetos, campos, classes, componentes, comportamentos do meu org ou requisitos que eu não tenha informado.
2. Separe claramente: FATO DO CONTEXTO, CONHECIMENTO GERAL, HIPÓTESE, PRECISA VALIDAR e PERGUNTA.
3. Se faltar contexto, faça perguntas antes de concluir.
4. Não trate uma hipótese como fato.
5. Antes de sugerir código, ajude-me a entender o fluxo atual, o impacto e a melhor forma de investigar.
6. Quando sugerir uma query, comando ou código, explique linha por linha ou bloco por bloco e diga o que devo esperar do resultado.
7. Faça perguntas para testar meu entendimento. Não faça tudo por mim.
8. Se houver mais de uma solução, compare alternativas, riscos e trade-offs.
9. Sempre indique o que preciso validar no ambiente real e na documentação oficial.
10. Se você não souber, diga que não sabe.

Meu objetivo é conseguir explicar a solução sozinho ao final.

Contexto bom versus contexto ruim

RUIM	BOM
"Crie o código dessa demanda."	"Eu li a demanda. Entendi A, B e C. Tenho dúvidas D e E. No ambiente encontrei os componentes X e Y. Ajude-me primeiro a validar o fluxo e montar um plano de investigação."
"Por que não funciona?"	"O erro acontece ao salvar. Entrada: X. Resultado esperado: Y. Resultado obtido: Z. Log relevante: [...]. Já validei A e B. Que hipóteses devo testar em ordem?"
"Faz uma SOQL."	"Preciso confirmar se registros do objeto X possuem Y quando Z. Estes campos existem: A, B, C. Monte uma SOQL somente de leitura e explique o filtro."
"Corrige meu Apex."	"Este método deveria fazer X. O teste falha no cenário Y. Seguem classe, teste e mensagem. Primeiro explique a causa provável e depois proponha a menor correção."

06

CAPÍTULO 06

Prompts práticos para o dia a dia

Os prompts abaixo são modelos para preencher com evidências do ambiente. Nenhum template substitui a investigação. Os trechos em [colchetes] são o que você completa.

PROMPT A Depois da leitura manual da demanda

Li a demanda manualmente e meu entendimento inicial é:

- Problema de negócio: [escreva]
- Comportamento atual: [escreva ou "ainda não sei"]
- Comportamento esperado: [escreva]
- Critérios de aceite identificados: [liste]
- Dúvidas que eu já levantei: [liste]

Quero que você:

1. Revise meu entendimento sem inventar informações.
2. Aponte ambiguidades e requisitos que parecem faltar.
3. Faça perguntas que eu deveria responder antes de pensar em solução.
4. Monte uma lista de coisas que preciso investigar no Salesforce.
5. Não proponha código ainda.

PROMPT B Montar o plano de investigação

Contexto confirmado até agora: [cole somente fatos e evidências].

Componentes que já encontrei: [liste].

O que ainda preciso descobrir: [liste].

Monte um plano de investigação em ordem, começando pelo caminho menos invasivo. Para cada passo, informe:

- objetivo;
- onde procurar;
- query/comando somente se necessário;
- qual evidência devo coletar;
- como interpretar o resultado;
- qual seria o próximo passo dependendo do resultado.

Não assumo que componentes não citados existem.

PROMPT C Gerar uma SOQL com aprendizado

Preciso investigar [objetivo].

Objeto confirmado: [API Name].

Campos confirmados: [lista].

Filtro de negócio: [condição].

Crie uma SOQL somente de leitura.

Depois explique cada cláusula, possíveis limitações, volume esperado e o que o resultado prova ou não prova.

Se faltar algum campo necessário, não invente: pergunte.

PROMPT D Analisar um payload OmniStudio

Estou analisando esta jornada OmniStudio.

Objetivo da etapa: [descreva].

Componente atual: [OmniScript/IP/Data Mapper/FlexCard/LWC].

Input real anonimizado: [cole].

Output real anonimizado: [cole].

Comportamento esperado: [descreva].

Compare input e output.

Separe o que é evidência do que é hipótese.

Aponte onde o valor pode estar sendo criado, transformado ou perdido, mas não invente nomes de elementos.

Sugira a próxima evidência que devo coletar.

PROMPT E Revisar Apex ou LWC

Este código é meu e eu preciso entendê-lo e melhorá-lo.

Objetivo funcional: [descreva].

Código: [cole].

Teste/erro: [cole].

Faça a revisão em quatro etapas:

1. Explique o fluxo do código em linguagem simples.
 2. Aponte problemas de lógica, segurança, limites, performance, legibilidade e testabilidade.
 3. Diferencie erro confirmado de melhoria opcional.
 4. Proponha a menor alteração necessária e explique por quê.
- Não reescreva tudo sem necessidade.

PROMPT F Criar casos de teste

Requisito confirmado: [descreva].

Regra de negócio: [descreva].

Componentes impactados: [liste].

Crie uma matriz de testes com:

- cenário;
- pré-condição;
- entrada;
- passos;
- resultado esperado;
- risco coberto.

Inclua cenário feliz, limites, nulos/vazios, permissões, regressão e erro de integração quando aplicável.

PROMPT G Quando você travou

Eu estou travado, mas não quero que você resolva por mim.

O que eu já sei: [liste].

O que já tentei: [liste].

Resultados obtidos: [liste].

Minha hipótese atual: [escreva].

Faça no máximo 5 perguntas de diagnóstico, em ordem de maior poder de eliminar hipóteses.

Depois de cada resposta minha, atualize a hipótese.

Não pule direto para código.

PROMPT H Fechar a demanda aprendendo

A demanda foi concluída. Quero consolidar o aprendizado.

Problema: [resuma].

Solução aplicada: [resuma].

Componentes alterados: [liste].

Testes: [liste].

Me faça 8 perguntas como se fosse um code review ou entrevista técnica para confirmar que eu realmente entendi.

Depois, gere um resumo de aprendizagem contendo conceito, decisão, erro que evitei, padrão reutilizável e ponto que ainda preciso estudar.

PROMPT I Transformar pendências em investigação objetiva

Estas são as minhas pendências atuais: [liste].

Estas regras já estão literalmente respondidas pelo requisito: [liste].

Estas evidências já coletei no ambiente: [liste].

Revise a lista e classifique cada item em:

- JÁ RESPONDIDO PELO REQUISITO;
- JÁ RESPONDIDO PELA EVIDÊNCIA;
- BUSCAR NA ORG;
- DÚVIDA FUNCIONAL;
- DECISÃO TÉCNICA.

Remova perguntas redundantes. Para cada item "BUSCAR NA ORG", transforme a dúvida em uma verificação concreta e diga qual evidência encerraria a questão. Não responda por inferência.

PROMPT J Rastrear uma cadeia OmniStudio/Salesforce

Quero rastrear um comportamento ponta a ponta sem assumir arquitetura.

Ponto inicial confirmado: [botão/tela/OmniScript/LWC/etc.].

Valor ou regra que estou seguindo: [descreva].

Componentes já confirmados: [liste].

Evidências atuais: [liste].

Ajude-me a montar a cadeia de investigação:

UI → orquestração → regra/cálculo → dados → persistência → consumidores → reprocessamento.

Para cada elo ainda desconhecido, diga exatamente o que pesquisar no VS Code/org/log/payload.

Não invente o próximo componente; formule a busca que pode encontrá-lo.

PROMPT K Escrever um script de investigação seguro

Preciso de um script para investigar [objetivo] em uma sandbox.

Objetos/campos confirmados: [liste].

Pré-condições conhecidas: [liste].

Quero que o script:

1. seja somente leitura sempre que possível;
2. valide pré-condições antes de continuar;
3. não invente API Names;
4. aborte com mensagem clara se faltar ID, Record Type, permissão ou campo necessário;
5. imprima resultados suficientes para eu entender o que aconteceu;
6. explique cada bloco e os riscos.

Se a investigação puder ser feita com SOQL ou metadata sem Apex anônimo, prefira a alternativa menos invasiva.

PROMPT L Limpar a lista de pendências antes de investigar

Vou te passar a demanda, a lista atual de pendências e o histórico de respostas. Para cada pendência, classifique apenas como:

RESPONDIDA PELO REQUISITO, RESPONDIDA POR EVIDÊNCIA, BUSCAR NA ORG, DÚVIDA FUNCIONAL, DECISÃO TÉCNICA, IMPACTO TÉCNICO ou DUPLICADA.

Antes de manter qualquer item aberto, mostre por que ele não está respondido. Se duas perguntas tratam da mesma coisa, diga qual deve permanecer e por quê. Não crie novas pendências sem apontar a lacuna concreta que as justifica.

PROMPT M Transformar "precisa validar" em investigação concreta

Você disse que isso precisa ser validado. Não pare aí.

Diga exatamente:

- o que validar;
- onde procurar;
- qual query/busca/metadata/log/payload/teste usar;
- qual evidência devo trazer;
- quais resultados confirmariam ou descartariam a hipótese.

Se precisar de API Name que eu não forneci, peça-o em vez de inventar.

P R O M P T N Fechar a análise antes do desenvolvimento

Com base apenas na demanda, nas decisões registradas e nas evidências que eu trouxe, monte um brief técnico com:

AS-IS confirmado, TO-BE confirmado, componentes alterados, impactos indiretos, fluxo de dados, regras/exceções, comportamento legado preservado, pendências restantes, riscos e matriz de testes.

Não proponha componente novo como fato se isso ainda for apenas decisão de implementação.

07

CAPÍTULO 07

Como usar IA para aprender enquanto desenvolve

A evolução acontece quando você é obrigado a recuperar e organizar o conhecimento, e não apenas reconhecer uma resposta pronta. Por isso, o uso da IA deve ter momentos em que ela explica e momentos em que ela devolve a responsabilidade para você.

Ciclo de aprendizagem em cinco movimentos

- 01 **Tente primeiro:** escreva sua hipótese antes de perguntar.
- 02 **Peça explicação:** solicite conceito, exemplo e relação com o seu caso.
- 03 **Faça você:** implemente ou formule a próxima etapa sem copiar automaticamente.
- 04 **Peça crítica:** mostre sua solução e peça problemas, alternativas e riscos.
- 05 **Ensine de volta:** explique com suas palavras e peça para a IA apontar lacunas.

REGRA DO COMMIT

Nunca faça commit de um trecho gerado por IA que você não consiga explicar. Se há uma linha que você não entende, pare nessa linha e estude antes de seguir.

Perguntas que desenvolvem raciocínio

- › Por que esta camada é melhor para colocar essa regra?
- › Que alternativa eu teria e qual seria o trade-off?
- › O que poderia quebrar se este campo vier nulo?
- › Qual governor limit pode ser relevante aqui?
- › Como este código se comporta com 1 registro e com 200 registros?
- › Como eu provaria que a causa do bug é realmente esta?
- › Que teste falharia se eu removesse esta parte da solução?
- › Qual evidência diferencia minha hipótese A da hipótese B?

Uma conversa por demanda, alimentada com contexto

Para uma demanda complexa, a conversa com a IA pode funcionar como um caderno técnico vivo. A ideia é registrar aos poucos o que foi descoberto, sem despejar o projeto inteiro, e corrigir o modelo quando uma hipótese cai.

- Comece com seu entendimento manual e as dúvidas iniciais.
- Ao descobrir um API Name, componente, payload ou comportamento real, atualize o contexto.
- Quando uma hipótese for negada pela org, diga explicitamente que ela está descartada.
- Peça periodicamente um resumo apenas dos fatos confirmados e das pendências ainda abertas.
- No final, peça um resumo do que você aprendeu e tente refazer mentalmente o fluxo sem consultar a conversa.

O GANHO DE APRENDIZAGEM ESTÁ NO CICLO

Você pergunta, investiga, volta com evidência, revisa a hipótese e explica o que entendeu. Copiar a primeira resposta pronta elimina justamente esse ciclo.

08

CAPÍTULO 08

Como validar código, configuração e arquitetura

A validação acontece em camadas. Nenhuma ferramenta isolada prova que a solução está correta.

CAMADA	PERGUNTA DE VALIDAÇÃO
Requisito	A solução atende exatamente ao que foi solicitado, inclusive exceções?
Ambiente	Os nomes, dependências e comportamentos foram confirmados na org?
Código / configuração	Compila, ativa e segue padrões do projeto?
Segurança	Respeita CRUD/FLS, sharing, contexto de execução, secrets e dados sensíveis?
Performance	Considera bulkificação, limites, volume, chamadas e consultas?
Testes	Há evidência de cenário feliz, negativo, limite e regressão?
Análise estática	Code Analyzer e linters apontam problemas relevantes?
Revisão humana	Mudanças críticas, arquiteturais ou pouco conhecidas foram revisadas por alguém experiente?

O Salesforce Code Analyzer pode entrar nessa rotina para análise estática de Apex, Visualforce, Flows e componentes Lightning, identificando problemas de segurança, performance e boas práticas. Ele complementa testes e revisão contextual, sem substituí-los.

09

CAPÍTULO 09

Testes, revisão e Definition of Done

Um júnior tende a parar quando "funcionou no meu teste". A disciplina profissional começa quando ele aprende a provar que a solução funciona, que não quebra cenários próximos e que os riscos conhecidos foram cobertos.

Checklist mínimo antes de dizer "terminei"

- ☐ Consigo explicar o requisito com minhas palavras.
- ☐ Sei quais componentes foram alterados e por quê.
- ☐ Sei quais componentes podem ser impactados mesmo sem alteração direta.
- ☐ Toda hipótese importante foi confirmada ou registrada como pendência.
- ☐ Não usei nomes, comportamentos ou regras inventados pela IA.
- ☐ Testei o cenário principal.
- ☐ Testei pelo menos um cenário negativo ou de limite relevante.
- ☐ Validei permissões e segurança quando aplicável.
- ☐ Revisei logs, payloads ou registros para confirmar o efeito real.
- ☐ Rodei os testes automatizados aplicáveis.
- ☐ Rodei análise estática quando aplicável.
- ☐ Comparei o resultado com o critério de aceite.
- ☐ Consigo explicar cada trecho de código que adicionei.
- ☐ Documentei decisões e pontos que precisam de atenção no deploy/QA.
- ☐ Se outra pessoa assumir amanhã, ela consegue entender o que foi feito.
- ☐ Antes de manter uma pendência aberta, procurei a resposta na demanda, no histórico e nas evidências já coletadas.
- ☐ Removi duplicidades e quebrei perguntas que misturavam mais de um assunto.
- ☐ Para todo "precisa validar", sei exatamente o que validar, onde e com qual evidência.
- ☐ Cada evidência registrada diz também o que ela NÃO prova.
- ☐ Tenho um brief de implementação que consigo explicar sem consultar a IA.

Teste comportamento, não apenas cobertura

Em Apex, cobertura de linhas é uma métrica necessária no processo de entrega, mas sozinha não demonstra que a regra está correta. Um teste útil prepara o cenário, executa o comportamento e verifica explicitamente o resultado esperado com asserts relevantes.

- Teste o caminho feliz e pelo menos os desvios realmente previstos pela regra.
- Teste erro, null/vazio, limite e permissão quando forem riscos reais da mudança.
- Para bugfix, primeiro reproduza o defeito e depois prove que o mesmo cenário foi corrigido.

- Para mudança em cálculo ou persistência, valide o valor antes, durante e depois da cadeia quando isso ajudar a localizar regressão.
- Teste também o comportamento legado que deveria permanecer igual quando a nova condição não se aplica.

Revisão da IA no final

No fim, peça uma revisão adversarial. O objetivo é tentar quebrar a sua própria solução antes que alguém quebre por você.

PROMPT Revisão adversarial

Atue como revisor técnico crítico. Tente encontrar formas de esta solução falhar.

Requisito: [cole o resumo confirmado]

Arquitetura atual: [resuma]

Alterações: [liste ou cole diff]

Testes executados: [liste]

Procure:

- requisitos não cobertos;
- regressões possíveis;
- valores nulos e dados inesperados;
- problemas de permissão e segurança;
- limites e volume;
- concorrência/ordem de execução;
- dependências não consideradas;
- testes ausentes.

Não invente componentes do meu org. Identifique cada ponto como CONFIRMADO PELO MATERIAL ou HIPÓTESE A VALIDAR.

10

CAPÍTULO 10

Segurança, dados e limites

Usar IA em ambiente corporativo exige cuidado com o conteúdo enviado. A conveniência nunca deve superar políticas da empresa, confidencialidade e proteção de dados.

Nunca envie sem necessidade e autorização

- ✗ senhas, tokens, API keys, cookies de sessão e secrets;
- ✗ secrets de Named Credential, client secrets e certificados privados;
- ✗ dados pessoais de clientes, pacientes, empregados ou terceiros;
- ✗ dados produtivos completos quando um exemplo anonimizado é suficiente;
- ✗ código proprietário ou documentação confidencial quando a política da empresa proibir;
- ✗ URLs internas, credenciais, IDs ou payloads que permitam identificar pessoas sem necessidade.

Prefira anonimizar valores, reduzir payloads ao mínimo necessário e substituir informações reais por marcadores consistentes. Se você precisa que a IA entenda o relacionamento entre registros, preserve a estrutura e esconda a identidade das pessoas.

SCRIPTS GERADOS POR IA

Queries de leitura também precisam ser revisadas. Scripts com INSERT, UPDATE, DELETE, UPSERT, Apex anônimo, alteração de metadata ou ações em produção exigem nível de revisão muito maior. Nunca execute uma ação destrutiva apenas porque a IA afirmou que é segura.

11

CAPÍTULO 11

Quando pedir ajuda humana

Autonomia não significa ficar horas preso. Saber quando escalar também é uma competência técnica. O objetivo é chegar ao sênior com a investigação feita, e não apenas com "não consegui".

Escale quando

- › o requisito é contraditório ou depende de decisão de negócio;
- › há mais de uma arquitetura possível e o trade-off envolve padrão do projeto;
- › a mudança toca integração crítica, segurança, dados sensíveis ou produção;
- › você não tem acesso suficiente para confirmar a hipótese;
- › o comportamento depende de pacote gerenciado ou sistema externo não documentado;
- › o impacto potencial é grande e não existe cobertura de testes adequada;
- › você repetiu experimentos e não conseguiu produzir nova evidência;
- › a IA e sua investigação indicam possibilidades diferentes, mas nenhuma pode ser comprovada.

Como pedir ajuda bem

MENSAGEM Estrutura de escalonamento

Problema: [1 a 3 linhas]
Comportamento esperado: [resumo]
Comportamento observado: [resumo]
O que já investiguei: [lista]
Evidências: [queries, logs, prints, componentes]
Hipóteses restantes: [lista]
O que preciso da sua ajuda para decidir/confirmar: [pergunta objetiva]

12

CAPÍTULO 12

Exemplo completo de uma demanda Salesforce

Exemplo fictício: uma user story pede que o vendedor possa informar descontos por faixa de quantidade antes de gerar uma cotação, numa jornada feita em OmniStudio. A regra informa quando a funcionalidade aparece e como os descontos devem influenciar o cálculo do total, mas não determina qual LWC, Integration Procedure, Apex ou estrutura de persistência deve ser alterada.

PRIMEIRO CUIDADO

"Precisa de uma nova grade/LWC" pode ser uma decisão técnica, e não um requisito funcional. O dev deve preservar literalmente o que a demanda exige e investigar o AS-IS antes de transformar sua primeira ideia de implementação em fato.

ETAPA 1

Leitura manual

TIPO	ITEM
FATO	A funcionalidade só aparece nas condições de negócio explicitadas pela demanda.
FATO	Os descontos informados precisam chegar ao cálculo e alterar o total da cotação.
FATO	Quando a nova opção não se aplica ou está desativada, o comportamento legado precisa permanecer.
AMBIENTE	Onde o desconto equivalente é calculado e usado hoje?
AMBIENTE	Onde a configuração atual é persistida e como entra em recálculo e reabertura da cotação?
DECISÃO ABERTA	Usar componente existente, novo LWC, Data Mapper, IP ou Apex depende do AS-IS e dos padrões do projeto.

ETAPA 2

Perguntas antes de desenvolver

- › Quais condições de negócio já estão literalmente definidas e não precisam virar dúvida?
- › Qual parte ainda é funcional e precisa ser confirmada com o analista?
- › Onde a configuração equivalente vive hoje e qual é sua granularidade?
- › Como o cálculo consome esse valor: por item, por faixa, por cotação ou agregado?
- › O que deve acontecer em edição, recálculo, reabertura e registros criados antes da mudança?

- Quais caminhos legados precisam continuar idênticos?

ETAPA 3

Investigação

- 01 Abrir a jornada atual e reproduzir o fluxo AS-IS antes de alterar qualquer coisa.
- 02 Rastrear o dado desde a tela até a orquestração, regra/cálculo, persistência e consumidores posteriores.
- 03 Inspecionar JSON/payload em pontos estratégicos para saber quando o valor nasce ou muda.
- 04 Buscar no projeto pelos API Names reais encontrados, em vez de imaginar nomes prováveis.
- 05 Usar SOQL, retrieve de metadata, logs e testes para responder pendências de ambiente.
- 06 Revisar a lista de dúvidas e eliminar tudo que já foi respondido por requisito ou evidência.
- 07 Registrar impacto em recálculo, histórico, caminhos legados, sync/async e componentes consumidores.

ETAPA 4

Uso da IA

Só agora o desenvolvedor entrega à IA os fatos observados e pede ajuda para comparar opções. Por exemplo: regra de visibilidade no próprio OmniScript versus valor calculado em camada anterior; persistência via Data Mapper versus Apex existente; impacto sobre reentrada na jornada; testes de null e de condição falsa.

ETAPA 5

Implementação incremental

- 01 Fazer a menor mudança que permita observar o novo dado na tela.
- 02 Validar input/output antes de adicionar persistência.
- 03 Adicionar persistência e testar isoladamente.
- 04 Testar a jornada completa.
- 05 Testar condição falsa, dado ausente, reentrada e regressão do fluxo antigo.

ETAPA 6

Fechamento

O desenvolvedor deve conseguir explicar: de onde vem a condição, onde ela é avaliada, por que a regra ficou nessa camada, onde o dado é persistido, quais componentes dependem dele, quais testes foram feitos e o que ainda é risco conhecido.

O método em uma linha

Requisito literal → Dúvidas classificadas → AS-IS provado → Cadeia rastreada → Pendências viram verificações → Evidências voltam à IA → Impacto mapeado → Menor solução coerente → Testes de regra e regressão → **Explicação final pelo próprio dev**

13

CAPÍTULO 13

Checklist de bolso

Quatro momentos, quatro perguntas para fazer a si mesmo. Imprima, cole do lado do monitor.

1

ANTES DA IA

Li tudo. Resumi com minhas palavras. Separei regra literal, dúvida de negócio, dúvida técnica e dúvida de ambiente. Retirei perguntas que a própria demanda já responde.

2

COM A IA

Dei contexto real. Proibi inferências. Pedi perguntas. Exigi explicação. Usei a IA para investigar e aprender, sem terceirizar a solução.

3

ANTES DE CODAR

Provei o AS-IS. Rastreei o fluxo e os dados ponta a ponta. Confirmei componentes, persistência, consumidores e caminhos legados. Sei o que é requisito e o que é decisão técnica.

4

ANTES DE ENTREGAR

Testei. Revisei. Rodei a análise aplicável. Validei o critério de aceite. Consigo explicar tudo que alterei.

14

CAPÍTULO 14

Trilha de evolução do júnior

A meta é reduzir aos poucos a dependência da IA para decisões simples e aumentar o uso dela em revisão, investigação avançada e comparação de alternativas.

FASE	COMPORTAMENTO ESPERADO	USO DA IA	SINAL DE EVOLUÇÃO
1. Orientado	Precisa de roteiro para investigar.	IA faz perguntas, explica conceitos e ajuda a montar consultas.	Para de pedir solução direta.
2. Investigador	Já chega com hipóteses e evidências.	IA critica o plano e sugere lacunas.	Consegue isolar a causa antes de codar.
3. Implementador consciente	Desenha e implementa partes pequenas.	IA revisa código, testes e trade-offs.	Explica cada decisão.
4. Autônomo em demandas comuns	Conduz o fluxo completo e escala apenas decisões relevantes.	IA acelera e desafia, sem conduzir o raciocínio básico.	Ajuda outro júnior a investigar.

Ritual semanal de 20 minutos

- ☐ Escolha uma demanda concluída e explique a arquitetura sem olhar o código.
- ☐ Liste uma coisa que a IA sugeriu e você comprovou que estava errada ou incompleta.
- ☐ Liste um conceito Salesforce novo que você aprendeu.
- ☐ Escolha um trecho de código/configuração e explique por que ele existe.
- ☐ Defina um assunto para estudar na semana seguinte a partir das lacunas reais encontradas.

OBJETIVO FINAL

O melhor uso da IA é ajudar o júnior a adquirir, de forma mais rápida e consciente, os hábitos de investigação, validação e raciocínio que o tornam progressivamente mais autônomo.

15

CAPÍTULO 15

Fontes oficiais e leituras recomendadas

As práticas deste guia combinam experiência de desenvolvimento com orientações oficiais sobre uso crítico de IA, prompting, qualidade de código e arquitetura Salesforce. Use as fontes abaixo para aprofundar e validar.

OpenAI Help Center: O ChatGPT diz a verdade?

Explica que modelos podem produzir respostas incorretas ou enganosas, inclusive com confiança, e recomenda verificar informações importantes em fontes confiáveis.

<https://help.openai.com/pt-br/articles/8313428-does-chatgpt-tell-the-truth>

Salesforce Trailhead: Prompt Fundamentals

Material oficial sobre como estruturar prompts com papel, contexto, tarefa, instruções, formato e restrições, além de testar e melhorar prompts.

<https://trailhead.salesforce.com/content/learn/modules/prompt-fundamentals>

Salesforce Trailhead: Generative AI Basics

Discute limitações da IA generativa, incluindo alucinações e a necessidade de verificar a correção factual do conteúdo gerado.

<https://trailhead.salesforce.com/content/learn/modules/generative-ai-basics>

Salesforce Developers: Salesforce Code Analyzer

Documentação oficial da ferramenta de análise estática. A versão atual usa o plugin Code Analyzer v5 e os comandos code-analyzer na Salesforce CLI; a v4 foi aposentada em 2025.

<https://developer.salesforce.com/docs/platform/salesforce-code-analyzer/guide/code-analyzer.html>

Salesforce CLI: data query

Referência oficial do comando para executar consultas SOQL via Salesforce CLI, inclusive com Tooling API.

https://developer.salesforce.com/docs/platform/salesforce-cli-reference/guide/cli_reference_data_query.html

Salesforce CLI: project retrieve start

Referência oficial para recuperar metadata de uma org para o projeto local.

https://developer.salesforce.com/docs/platform/salesforce-cli-reference/guide/cli_reference_project_retrieve_start.html

Salesforce Well-Architected Framework

Estrutura oficial para avaliar a qualidade de soluções Salesforce sob perspectivas como trust, reliability, operational excellence e outros pilares.

<https://architect.salesforce.com/docs/architect/well-architected/guide/framework.html>



Leia primeiro. Pense antes. Investigue o ambiente. Use a IA para aprender e desafiar seu raciocínio. Valide tudo que importa. Só entregue aquilo que você consegue explicar.

MONIQUEDEV