CASE · DESIGN DE PROCESSO & AI WORKFLOW

A parte difícil não foi construir. Foi *descobrir o que estava errado* na pergunta.

Como projetei uma skill de prototipagem de alta fidelidade assistida por IA para um time de UX — e por que o trabalho real foi derrubar, uma a uma, as premissas com que eu comecei.

PAPEL	CONTEXTO	FORMATO	FOCO DO CASE
Product Designer / líder de workflow	Consultoria de tecnologia	Skill + design system + spec de handoff	Tomada de decisão sob incerteza

O PONTO DE PARTIDA

Um pedido aparentemente simples de "atualizar" uma ferramenta.

O time de UX já usava uma skill — um protocolo estruturado que orienta uma IA a transformar um fluxo de UX em telas de alta fidelidade e gerar a documentação de handoff para desenvolvimento.

Surgiram duas novidades: uma nova ferramenta de design assistida por IA, e um design system próprio recém-criado. O pedido inicial era intuitivo: **incorporar as duas coisas à skill existente.** Parecia uma atualização incremental — juntar o que havia de novo ao que já funcionava.

Se eu tivesse tratado como atualização incremental, teria construído a coisa errada. O valor deste projeto está inteiramente no que aconteceu quando parei de aceitar a premissa e comecei a testá-la.

O TRABALHO REAL · CINCO VIRADAS

Cada premissa que eu não questionasse viraria um defeito embutido na ferramenta.

A skill seria usada por um time inteiro, repetidamente, para entregar a clientes. Um erro de premissa não seria um erro meu — seria multiplicado por todas as pessoas e todos os projetos que usassem a ferramenta. Então cada suposição precisou ser derrubada antes de virar código.

01

PREMISSA INICIAL

"A nova ferramenta de IA substitui o editor de design de sempre."

O QUE DESCOBRI

A ferramenta nova é excelente para ideação, mas não substitui o editor tradicional em governança de componentes, tokens e handoff. Assumir substituição teria quebrado justamente a etapa mais importante para o desenvolvedor. **Ela adiciona uma etapa; não troca o motor.**

02

PREMISSA INICIAL

"Temos um design system pronto para governar o handoff."

O QUE DESCOBRI

Ao inspecionar o material, o "design system" era, na verdade, uma **camada de marca** — tokens, cores, tipografia, gradientes — **sem nenhum componente real**. Riquíssimo como skin; incapaz de sustentar um handoff por componente. Nomear isso corretamente mudou toda a arquitetura seguinte.

03

PREMISSA INICIAL

"A skill vai orquestrar a ferramenta de IA de ponta a ponta."

O QUE DESCOBRI

A skill roda num ambiente que **não controla** a ferramenta de ideação — são contextos separados. A instrução que eu havia escrito mandava a skill fazer algo que ela não tinha como executar. Definir com precisão *onde a ferramenta vive e o que ela alcança* evitou um travamento na primeira execução real.

04

PREMISSA INICIAL

"Camada de marca e biblioteca de componentes competem entre si."

O QUE DESCOBRI

Elas **se compõem**: estrutura (o que o componente é) vem de uma base de componentes; skin (como ele parece) vem da marca. A skill passou a separar as duas camadas explicitamente, com uma árvore de decisão simples — marca do cliente quando existe, base + skin própria como padrão da casa — em vez de negociar caso a caso.

05

PREMISSA INICIAL

"Fazer uma tela só é uma versão mais simples de fazer um fluxo."

O QUE DESCOBRI

É o oposto: a tela isolada é o caso **mais arriscado**. Ela precisa se encaixar num protótipo que já existe, herdando componentes, nomenclatura e convenções que a ferramenta não vê sozinha. O modo "tela única" exigiu mais salvaguardas que o modo "fluxo", não menos.

A DECISÃO DA QUAL MAIS ME ORGULHO

Recusar o atalho que parecia pronto.

Quando a ferramenta não consegue ler o arquivo-fonte de um protótipo existente, o plano de contingência era reconstruir a tela a partir de capturas de imagem. Fácil, rápido — e perigoso: gera um entregável com **aparência de pronto e qualidade de rascunho**, porque componentes e tokens passam a ser inferidos de pixels.

Em vez de deixar isso passar silenciosamente, projetei o modo de contingência para se **autodenunciar**: a ferramenta avisa na hora que a fidelidade caiu, marca a tela como provisória e imprime, no topo do documento, um alerta de que aquilo precisa ser reconciliado antes do desenvolvimento. A ferramenta assume a própria limitação em vez de escondê-la.

PRINCÍPIOS QUE GUIARAM O DESIGN

O que eu otimizei, além de "funcionar".

Honestidade sobre limites

Uma ferramenta que finge saber mais do que sabe produz erros silenciosos. Cada ponto de incerteza é sinalizado ao usuário, não escondido.

Rigor onde importa

O handoff para desenvolvimento é inegociável e sempre gerado. Etapas que não agregam a um caso específico são cortadas — sem cerimônia por cerimônia.

Consistência acima de velocidade

Uma tela nova precisa ser indistinguível das vizinhas. A ferramenta herda o contexto do que já existe antes de criar algo novo.

Feito para um time, não para mim

Como muitas pessoas usariam a mesma ferramenta, ambiguidade vira defeito multiplicado. Regras explícitas > julgamento improvisado a cada uso.

O ANTES E O DEPOIS

Como a ferramenta mudou de forma.

SKILL ORIGINAL

- Uma base de componentes fixa como padrão visual
- Relação marca × componentes negociada caso a caso
- Um único modo: fluxo inteiro
- Assumia construir tudo do zero, num só ambiente
- Marca aplicada "de memória", sem fonte única

SKILL REDESENHADA

- → Estrutura e skin como camadas separadas e explícitas
- → Árvore de decisão fixa: cliente, ou padrão da casa
- → Dois modos: fluxo e tela única, com processos distintos
- → Ciente de onde roda e do que alcança em cada ambiente
- → Marca embutida como fonte única, sem dependência externa

O QUE LEVO DESTE PROJETO

Aprendizados transferíveis para qualquer produto.

1 **A primeira versão do problema quase nunca é a verdadeira.** "Atualize isto" virou "redefina o que isto é". Boa parte do design é reescrever o brief antes de executá-lo.

2 **Nomear as coisas com precisão é uma decisão de design.** Chamar uma camada de marca de "design system" teria contaminado toda a arquitetura. Vocabulário correto evita erro estrutural.

3 **Projetar para o modo de falha, não só para o caminho feliz.** O que a ferramenta faz quando *não* consegue fazer o certo definiu a qualidade dela mais do que o fluxo ideal.

4 **Ferramenta de time amplia decisões.** Uma escolha ambígua não afeta um projeto — afeta todos. Isso eleva o custo de cada premissa não testada.

5 **Saber o que fica em aberto é entregar melhor.** Fechei o projeto explicitando os riscos que dependem de comportamento humano e de um teste ainda não feito — em vez de fingir que estava tudo resolvido.

Case documentado a partir do processo real de design da ferramenta.

Nomes de empresa e clientes omitidos por confidencialidade.