Projeto tabela FIPE-consultas

Sumário:

Eu sou o time inteiro e preciso agir como empresa: planejar → documentar → prototipar → codar → testar → apresentar.

• Critério implícito: clareza de arquitetura + organização do trabalho + escopo mínimo funcionando (não quantidade de features).

1) Planning da semana

.....

2) Modelo de Dados (SQLite)

Decisão chave: preço de referência é derivado em tempo de consulta a partir de pesquisa_preco. A cotação mensal consolidada pode existir em um sistema real (batch), mas aqui fica como extensão projetada.

Entidades mínimas para o MVP (implementado)

Tabela	Campos principais	Observação
marca	id, nome	Catálogo
modelo	id, marca_id, nome	Filtra por marca
versao	id, modelo_id, nome, ano_modelo	Ano-modelo/versão
pesquisa_preco	id, versao_id, data_coleta, preco, (loja_id opcional)	Dado bruto usado para cálculo
consulta_log (opcional)	id, dt, mes_ref, ano_ref, versao_id, mes_usado, ano_usado, preco_resultado	Prova de uso (simples)

Entidades do sistema completo (projetadas)

Papel/Entidade	Responsabilidade no sistema completo
Admin	Gerir usuários e perfis (se existisse login)
Gerente	Manter catálogo global (marca/modelo/versão/atributos)
Coordenador	Definir plano semanal de lojas por região; aprovar cadastros
Lojista	Cadastrar loja (com aprovação)
Pesquisador	Coletar e registrar preços nas lojas
Região/Loja/Agenda semanal	Organizar coleta e cobertura geográfica
Batch mensal	Consolidar médias do mês, validar qualidade e fechar referência

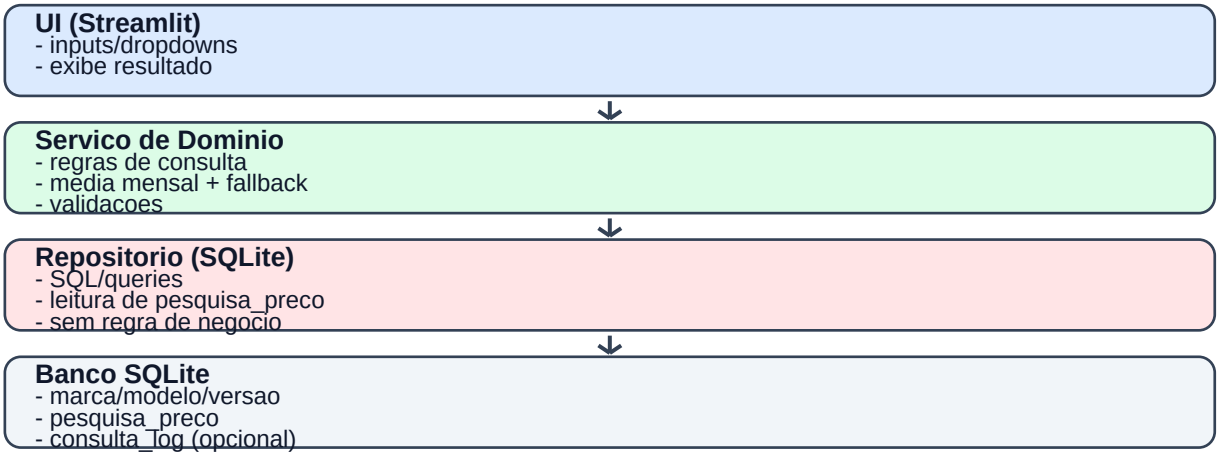
Regras de cálculo (domínio)

• Entrada: (mes_ref, ano_ref, versao_id).
• Passo 1: buscar pesquisas em pesquisa_preco dentro do mes_ref/ano_ref.
• Passo 2: se existir, retornar media(preco).
• Passo 3: se não existir, buscar o último mês anterior com dados para aquela versão.
• Passo 4: retornar media(preco) do mês encontrado e informar mes_usado/ano_usado.
• Passo 5: se não existir mês anterior, retornar NOT_FOUND (sem cotação disponível).

3) Modelo de Componentes (Monólito com separação de responsabilidades)

Objetivo: manter UI simples e trocar regras/banco sem reescrever tudo. A UI não faz SQL; o repositório não decide regra; o domínio orquestra a consulta.

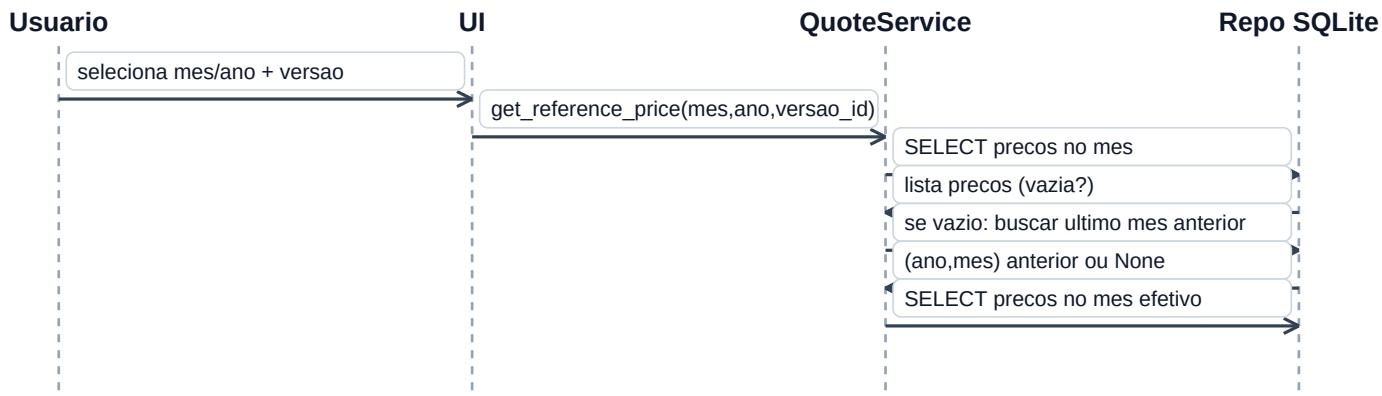
Camadas



Contratos principais (interfaces)

Componente	Assinatura (exemplo)	Responsabilidade
CatalogService	list_brands(); list_models(marca_id); list_versions(modelo_id)	Fornecer opções para dropdowns
QuoteService	get_reference_price(mes, ano, versao_id) -> QuoteResult	Aplicar média + fallback + mensagens
PriceResearchRepository	get_prices_for_month(...); find_latest_month_before(...)	Executar queries no SQLite
LogRepository (opcional)	insert_query_log(...)	Registrar consultas (simples)

Sequência — Consulta com fallback



Próximos passos imediatos (quarta)

• Criar schema.sql (SQLite) e seed_db.py com datas que provem o fallback.
• Implementar UI Streamlit mínima (dropdowns + botão Consultar + resultado).
• Implementar QuoteService com regras e mensagens.
• Adicionar 3 testes básicos (CT01-CT03) como prova de qualidade.