

INSTITUTO MINERVA



Projeto Tabela FIPE – Consultas

Documentação de planejamento, modelo de dados e modelo de componentes

Alexsandro Pereira • 10/02/2026

Sumário

- Planning
- Modelo de Dados
- Modelo de Componentes (TDD resumido).

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

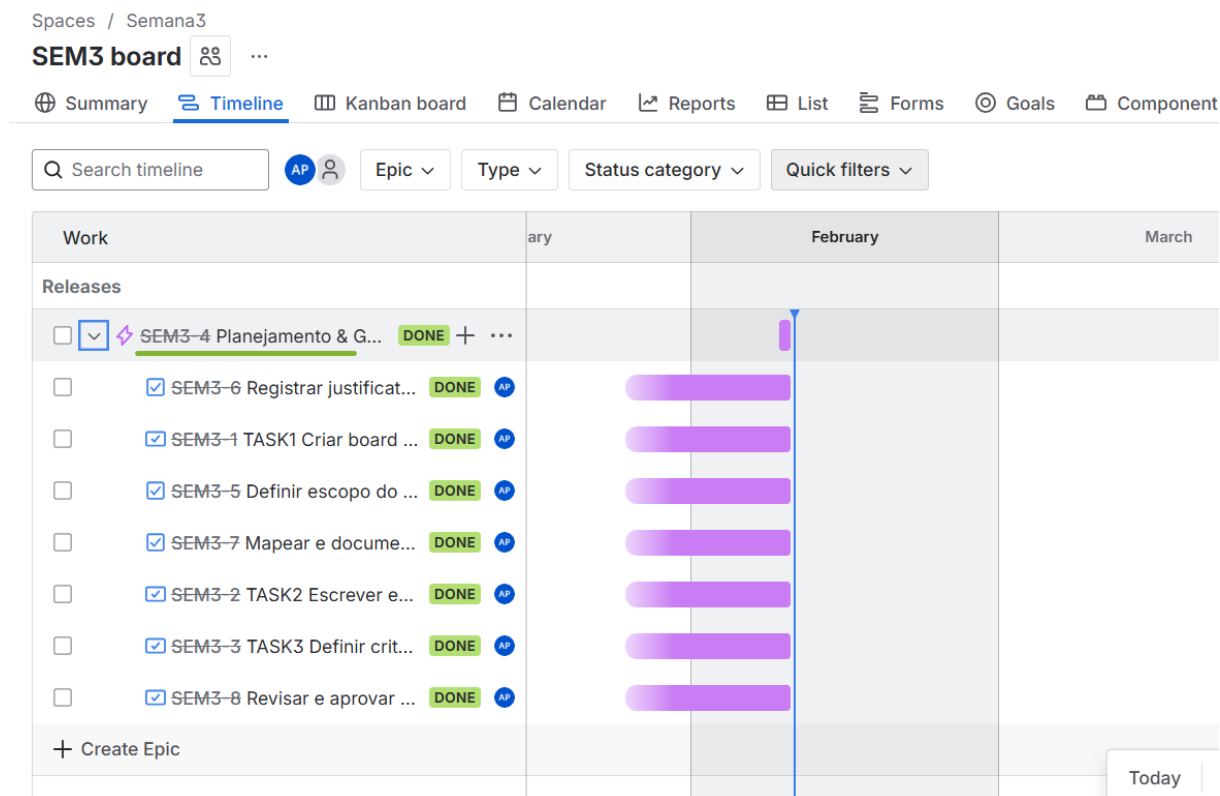
- Critério implícito: clareza de arquitetura + organização do trabalho + escopo mínimo funcionando (não quantidade de features).

1) Planning da semana

Objetivo do MVP: disponibilizar uma interface de consulta de preços (estilo FIPE) com seleção por *Marca* → *Modelo* → *Ano/Versão*, retornando **preço** e **mês de referência**. Requisito adicional: registrar um log simples de consultas para análise posterior.

Segunda	Planning	Backlog, escopo (in/out), critérios de aceite, riscos e plano de execução
Terça	Documentação de projeto	Arquitetura, modelo de dados (ERD) e TDD/Componentes (resumo)
Quarta	Prototipação + início de código	Protótipo da UI, esqueleto do projeto e seed do banco
Quinta	Codificação + testes	Fluxo de consulta completo, fallback de mês e testes unitários básicos
Sexta	Integração + apresentação	Polimento, revisão final, evidências (prints), roteiro e demo

Evidências de planejamento (board e cronograma)



Projeto FIPE-like — Semana Final (Solo)

Entregável de hoje (terça): Planning + Modelo de Dados + Modelo de Componentes (TDD resumido).

Contexto e mindset

- Semana 2 (time): módulos divididos (front, regras, BD, API), entregas contínuas em sprints.
- Semana final (solo): você é o time inteiro e precisa 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 (alto nível)

Dia	Foco	Entrega (artefatos)
Terça	Planning + Documentos	Escopo (funcional), Modelo de Dados, Modelo de Componentes/TDD
Quarta	TDDs + Protótipos + Início do código	TDD consulta + sequência; protótipo UI; schema SQLite + seed; primeira ver
Quinta	Codar + Testar + Refinar	Consulta funcionando com fallback; logs simples; testes básicos; ajustes UX
Sexta	Consolidar + Apresentar	Demo estável + relatório final (decisões, trade-offs, backlog, próximos passo:

Backlog (Kanban mental) e rastreabilidade

Colunas sugeridas: Backlog → Ready → Doing → Review → Done. Exemplos de cards já coerentes com o plano:

Backlog	Ready	Doing	Review	Done
docs/01_escopo.md	docs/02_modelo_dados.md docs/03_tdd_design.md	schema.sql seed_db.py UI consulta	fallback ultimo mes testes CT01-CT03	demo + pitch

Definition of Done (DoD) — Consulta do Usuário

- Dropdowns dependentes: Marca → Modelo → Versão/Ano-modelo.
- Consulta calcula média do mês a partir de pesquisa_preco.
- Se não houver dados no mês escolhido, aplica fallback para o último mês anterior disponível e informa mês/ano usado.
- Se não houver dados em nenhum mês anterior, retorna mensagem clara.
- Seed mínimo para demo (dados suficientes para provar o fallback).
- Código organizado por camadas (UI / domínio / repositório).

Projeto FIPE-like — Semana Final (Solo)

Entregável de hoje (terça): Planning + Modelo de Dados + Modelo de Componentes (TDD resumido).

Contexto e mindset

- Semana 2 (time): módulos divididos (front, regras, BD, API), entregas contínuas em sprints.
- Semana final (solo): você é o time inteiro e precisa 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 (alto nível)

Dia	Foco	Entrega (artefatos)
Terça	Planning + Documentos	Escopo (funcional), Modelo de Dados, Modelo de Componentes/TDD
Quarta	TDDs + Protótipos + Início do código	TDD consulta + sequência; protótipo UI; schema SQLite + seed; 1ª versão da UI
Quinta	Codar + Testar + Refinar	Consulta funcionando com fallback; logs simples; testes básicos; ajustes UX
Sexta	Consolidar + Apresentar	Demo estável + relatório final (decisões, trade-offs, backlog, próximos passos)

Backlog (Kanban mental) e rastreabilidade

Colunas sugeridas: Backlog → Ready → Doing → Review → Done. Exemplos de cards já coerentes com o plano:

Backlog	Ready	Doing	Review	Done
docs/01_escopo.md	docs/02_modelo_dados.md	schema.sql	fallback ultimo mes	demo + pitch
	docs/03_tdd_design.md	seed_db.py	testes CT01-CT03	
		UI consulta		

Definition of Done (DoD) — Consulta do Usuário

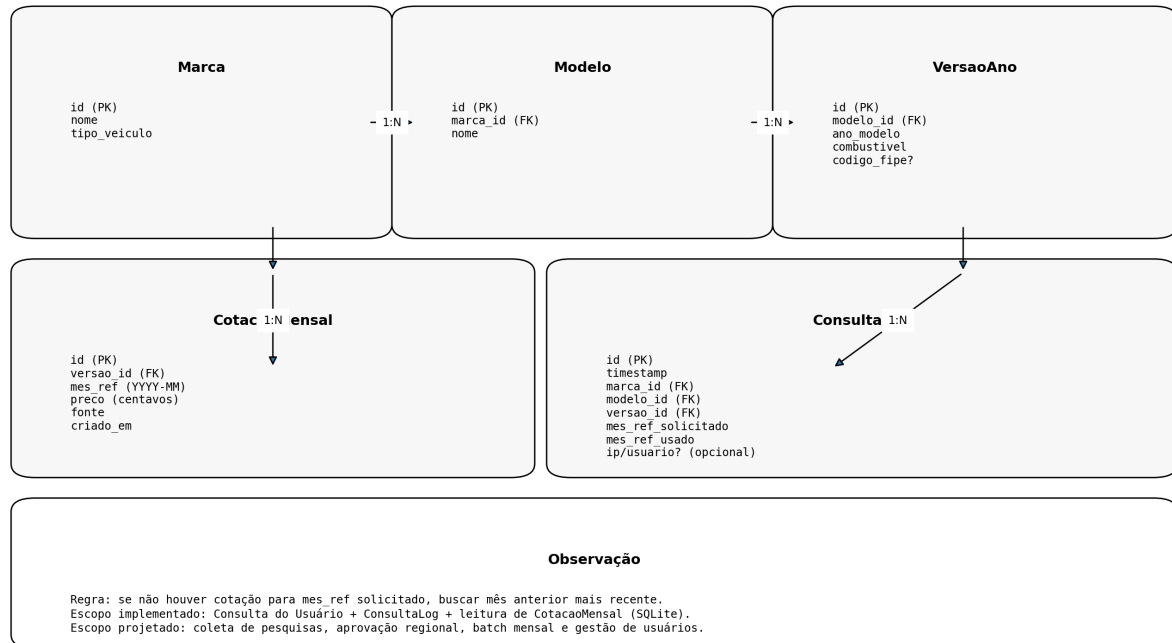
- Dropdowns dependentes: Marca → Modelo → Versão/Ano-modelo.
- Consulta calcula média do mês a partir de pesquisa_preco.
- Se não houver dados no mês escolhido, aplica fallback para o último mês anterior disponível e informa mês/ano usado.
- Se não houver dados em nenhum mês anterior, retorna mensagem clara.
- Seed mínimo para demo (dados suficientes para provar o fallback).
- Código organizado por camadas (UI / domínio / repositório).

Critérios de qualidade e rastreabilidade

- Escopo explícito (In/Out): a consulta do usuário é o único item implementado; demais papéis permanecem no projeto/documentação.
- Critérios de aceite: a consulta retorna preço e mês; aplica fallback para o último mês disponível quando a referência solicitada não existir.
- Definition of Done: código executável, dados seedados, testes básicos, log de consulta e documentação atualizada.
- Rastreabilidade: decisões relevantes registradas como tickets/notas vinculadas aos épicos correspondentes.
- Qualidade do desenho: ERD e componentes revisados para legibilidade (chaves, relações e responsabilidades).

2) Modelo de Dados

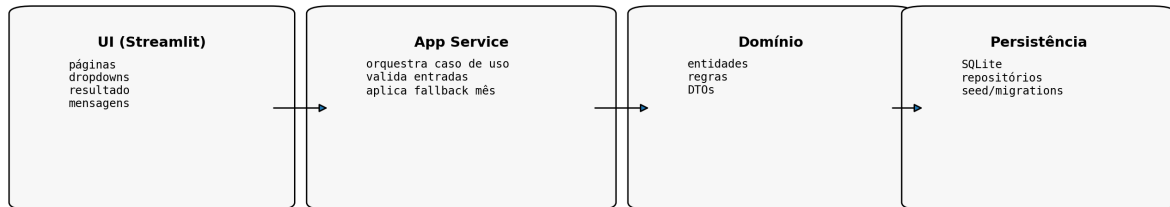
O modelo de dados abaixo suporta o sistema completo (papéis operacionais e batch mensal), mantendo o MVP focado na consulta do usuário.



Regra de fallback de mês: ao consultar uma referência inexistente, buscar a última referência disponível anterior para o mesmo veículo (marca/modelo/versão).

3) Modelo de Componentes (TDD resumido)

Arquitetura simples (monólito com separação de responsabilidades) para permitir evolução do domínio sem acoplamento excessivo à UI.



Separação de responsabilidades: UI não conhece SQL; regras ficam no serviço/domínio.

Responsabilidades (resumo): UI coleta filtros e exibe resultado; Serviço de Consulta aplica regra de fallback e orquestra repositórios; Repositórios encapsulam acesso ao SQLite; Logger registra consultas.