

# Projeto tabela FIPE-consultas

Kumbem

11/02/2026 — Entrega 2

## 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).

### Escopo mínimo implementável (o que será codado):

- Interface de consulta (usuário final), estilo FIPE:
  - Dropdown 1: **Marca**
  - Dropdown 2: **Modelo**
  - Dropdown 3: **Ano-modelo** / **versão**
  - Botão **Consultar**
  - Resultado: **preço** + **mês de referência**
- Log simples de consulta (registrar que houve consulta).
- Regra de fallback: **se não existir cotação no mês escolhido, retornar o último mês disponível**.

**Fora do escopo de código (apenas projetado/documentado):** Admin / Gerente / Coordenador / Pesquisador / Lojista / Batch mensal (fechamento).

# 1) 1) Planning da semana

## 1.1 1.1 Board (evidência de organização)

Figura abaixo: visão do Jira (*SEM3 board*) com tarefas de planejamento concluídas e rastreabilidade do trabalho.

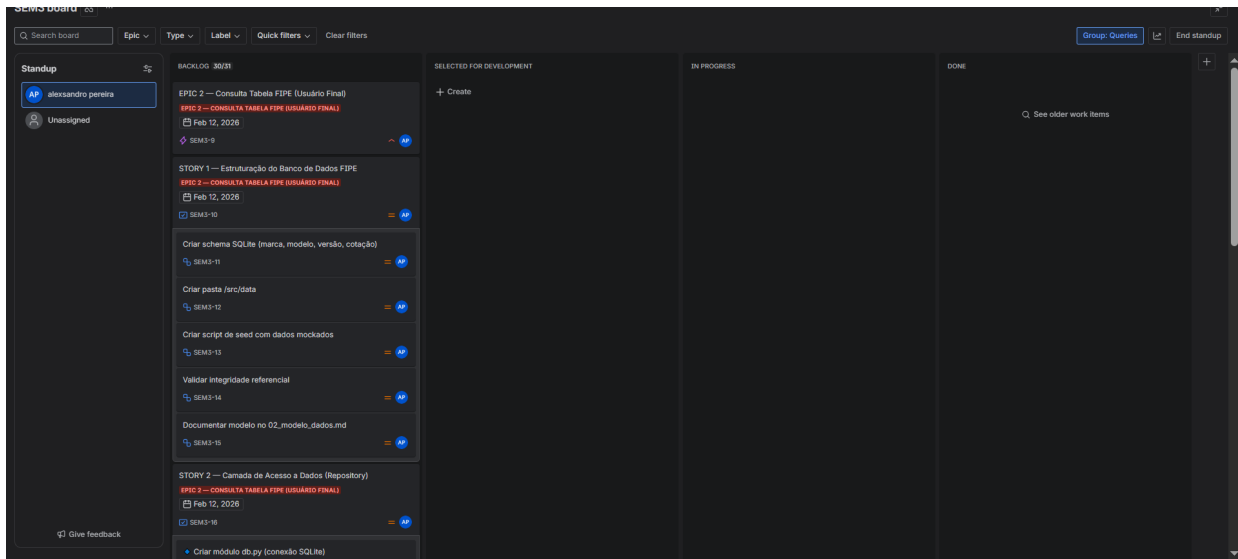


Figura 1: Jira SEM3 — organização da planning.

## 1.2 1.2 Board da semana

A semana é planejada para maximizar **clareza de arquitetura + entrega mínima funcional** dentro do tempo.

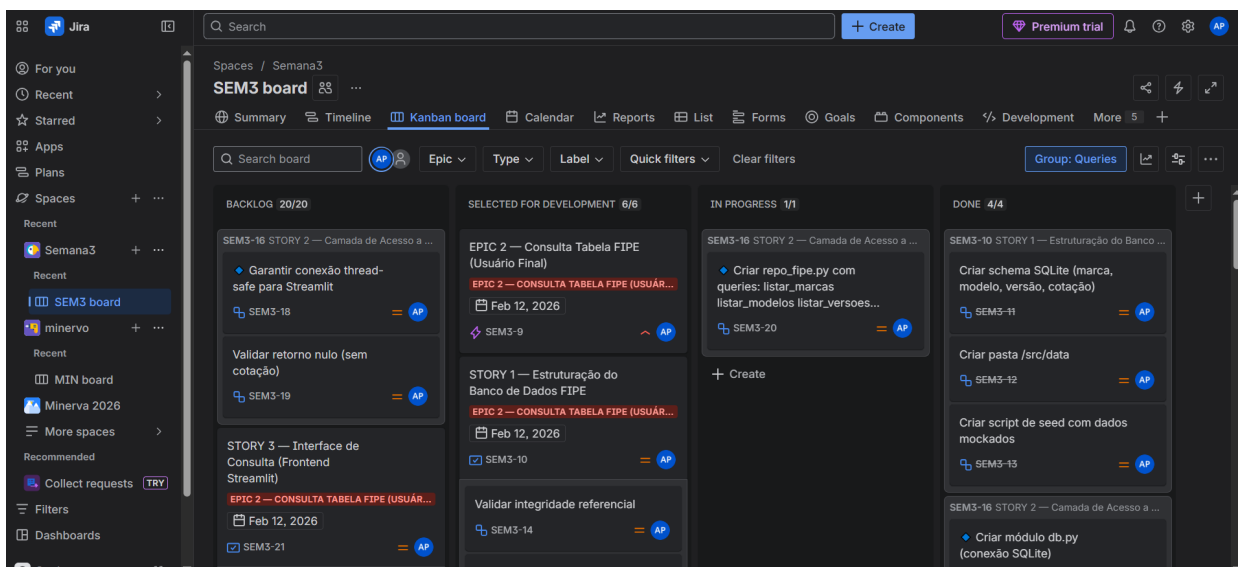


Figura 2: Plano semanal (macro)

## 1.3 1.3 Entregas e DoD (Definition of Done)

### Entrega 2:

- **OK** Documento com: Planning + Modelo de Dados + Modelo de Componentes (TDD resumido).

### DoD do MVP :

- **OK** Tela Streamlit com 3 dropdowns em cascata + botão Consultar.
- **OK** Retorna preço + mês de referência.
- **OK** Aplica fallback do último mês disponível quando faltar cotação no mês selecionado.
- **OK** Registra log simples da consulta.
- Código organizado em camadas (UI / Service / Repository).

## 2 2) Decisões explícitas (Mini ADRs)

### ADR-001 — Persistência

**Decisão:** usar **SQLite** com **seed local**.

**Motivo:** velocidade de implementação + simplicidade + rastreabilidade do dado.

**Consequência:** dados reais de operação não existem; simulação via seed.

### ADR-002 — Arquitetura

**Decisão:** monólito em camadas (**UI** → **Service** → **Repository**).

**Motivo:** fácil de avaliar e evoluir (troca regra sem mexer na UI; troca BD sem quebrar regra).

**Consequência:** sem microserviços; foco na clareza.

### ADR-003 — Regra de consulta com fallback

**Decisão:** se mês selecionado não tiver cotação, retornar **último mês disponível** para a mesma versão/ano.

**Motivo:** comportamento realista e robusto (dados podem estar incompletos).

**Consequência:** precisa de query por “máximo mês disponível”.

### 3) Rastreabilidade mínima (Requisito → Entidade → Componente)

Requisito		Entidades (dados)	Componentes (código)
Consulta (MVP)	FIPE	Marca, Modelo, VersaoAno, CotacaoMensal	UI(Streamlit), QueryService, SQLiteRepo
Fallback mês	último	CotacaoMensal	QueryService (regra) + SQLiteRepo (query ordenada por mês)
Log simples de consulta		ConsultaLog	QueryService (orquestra) + SQLiteRepo (insert log)

Tabela 1: Rastreabilidade do MVP (evidência de pensamento de liderança técnica).

## 4) Modelo de Dados (ERD)

### 4.2 Diagrama ER (com destaque do MVP)

**MVP** Entidades do MVP: Marca, Modelo, VersaoAno, CotacaoMensal, ConsultaLog.

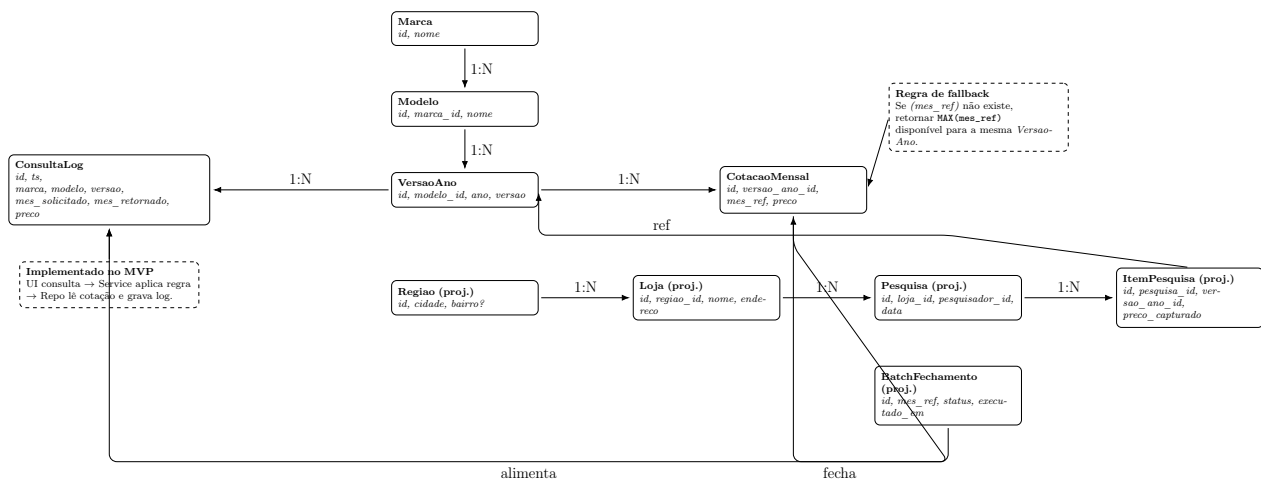


Figura 3: ERD completo (projetado) com foco no **MVP** e regra de fallback.

## 5) Modelo de Componentes (TDD resumido)

### 5.1 Camadas e responsabilidades

- **UI (Streamlit):** captura entradas e renderiza saída. Sem regra de negócio.
- **Service (Domínio):** valida seleção, aplica fallback do mês, orquestra log.
- **Repository (SQLite):** queries e persistência (cotação e log).

## 5.2 5.2 Contratos (interfaces) — versão mínima

- **QueryService**

- `list_marcas()`
- `list_modelos(marca_id)`
- `list_versoes(modelo_id)`
- `consultar(versao_id, mes_solicitado) → (preco, mes_retornado)`

- **SQLiteRepo**

- `get_cotacao(versao_id, mes)`
- `get_ultima_cotacao(versao_id, mes_limite)`
- `insert_log(...)`

## 5.3 5.3 Sequência do fluxo (consulta)

1. Usuário escolhe Marca → Modelo → Versão/Ano e um mês.
2. UI chama `QueryService.consultar()`.
3. Service tenta `Repo.get_cotacao()`.
4. Se não existir: Service chama `Repo.get_ultima_cotacao()` e retorna o último mês disponível.
5. Service registra `ConsultaLog`.
6. UI exibe preço + mês retornado.

## 6 6) Critérios de aceite do MVP (3 bullets)

- Dropdowns em cascata (marca → modelo → versão/ano) com estado consistente.
- Consulta retorna **preço + mês de referência**.
- Na ausência do mês solicitado, retorna **último mês disponível** e registra log.

## 7 7) Plano de teste mínimo

### Testes essenciais (manual + 1 automatizado)

- **Happy path:** existe cotação no mês escolhido → retorna preço correto e log gravado.
- **Fallback:** mês solicitado sem cotação → retorna último mês disponível + log com `mes_retornado`.
- **Dados incompletos:** marca sem modelos (ou modelo sem versões) → UI não quebra, mostra mensagem clara.
- **Automatizado (unit):** teste do Service para fallback (mock do repo).

---

*Entrega 1 foca em: clareza + organização + MVP consultável e robusto.*