Minerva

Instituto Minerva
Residência em Engenharia

Sidevaldo Vinícius Paulino de Souza

Russas - CE 2026

Minerva

SUMÁRIO

Link para o jira: https://sidevaldopaulino.atlassian.net/jira/software/projects/S3M/boards/67

Detalhamento de conteúdo funcional:

A modelagem de dados apresentada no próximo tópico foi derivada diretamente das Histórias de Usuário. A necessidade de flexibilidade nos 'Opcionais' descrita na US06 justificou a escolha do tipo de dado JSONB no PostgreSQL.

US01 - Manter Usuários (Admin): "Como Administrador, quero cadastrar novos usuários e definir seus perfis (Gerente, Coordenador, Pesquisador), para garantir o acesso seguro ao sistema."

US02 - Gerir Marcas e Modelos (Gerente): "Como Gerente, quero cadastrar marcas e modelos de veículos globalmente, para padronizar as opções disponíveis nas pesquisas."

US03 - Alocar Pesquisas (Coordenador): "Como Coordenador Regional, quero definir semanalmente quais lojas cada Pesquisador deve visitar, para organizar a força de trabalho."

US04 - Aprovar Novas Lojas (Coordenador): "Como Coordenador, quero analisar documentos de novas lojas cadastradas e aprovar ou rejeitar sua inclusão na base."

US05 - Cadastrar Loja (Lojista/Pesquisador): "Como Lojista ou Pesquisador, quero cadastrar os dados básicos de uma loja, para que ela possa receber visitas de coleta de preços."

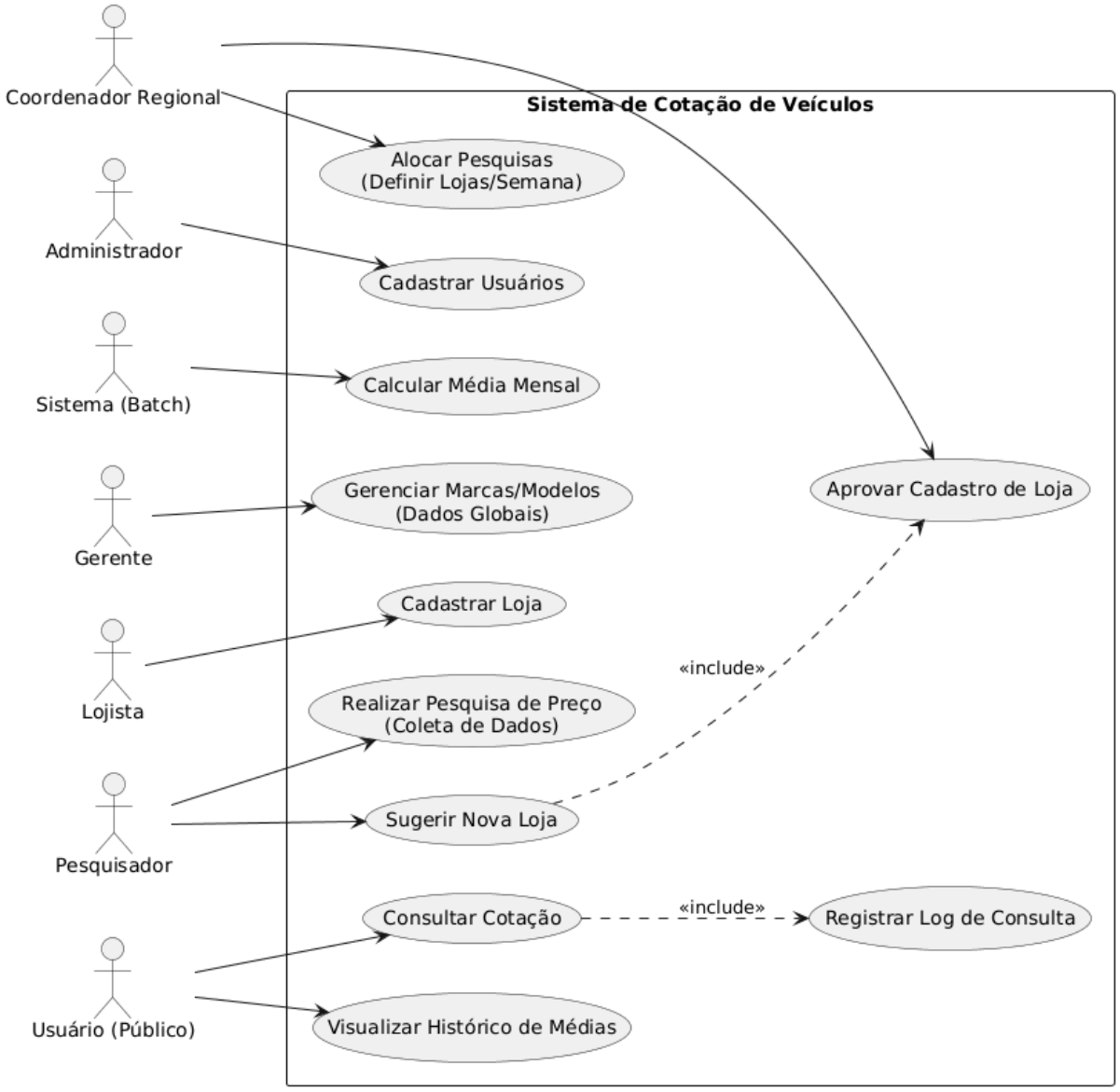
US06 - Coletar Preços (Pesquisador): "Como Pesquisador, quero registrar os preços, ano e opcionais dos veículos nas lojas alocadas para mim, para alimentar a base de dados do sistema."

US07 - Consultar Cotação (Usuário Final): "Como Usuário, quero pesquisar preços de veículos filtrando por marca, modelo e ano, para saber o valor de mercado atual."

US08 - Visualizar Histórico de Preços (Usuário Final): "Como Usuário, quero ver um gráfico com a média de preço mensal de um veículo, para entender a tendência de valorização ou desvalorização."

US09 - Calcular Médias Mensais (Sistema): "Como Sistema (Processo Batch), quero calcular automaticamente a média de preços de todos os modelos uma vez por mês, para otimizar a geração de relatórios históricos."

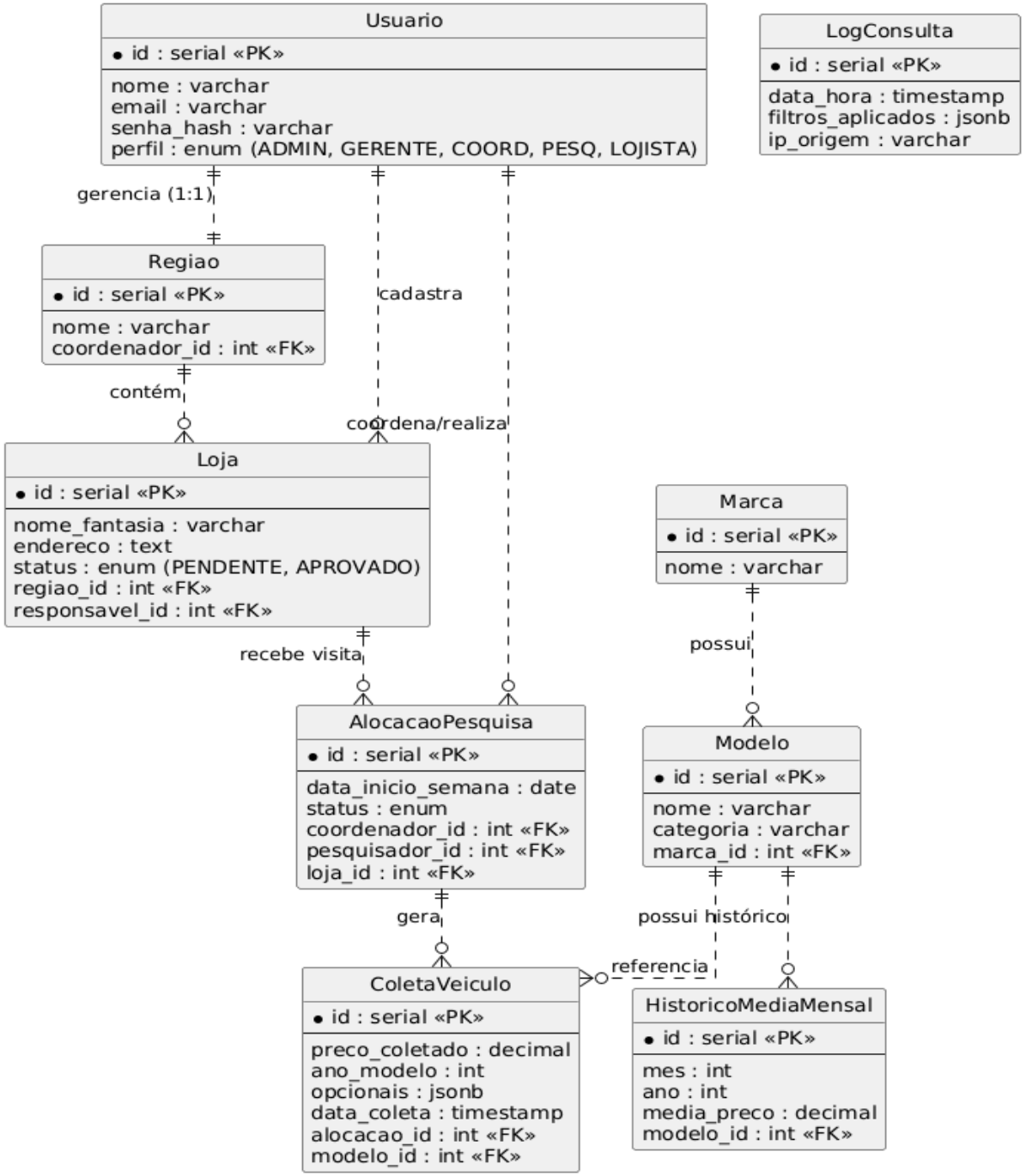
- Diagrama de caso de uso:



Minerva

Modelo de Dados (SQL) :

- Diagrama Entidade-Relacionamento:



Minerva

Estrutura das tabelas e construção da entidade-relacionamento.

(Em 3ºF normal).

```
CREATE TYPE tipo_perfil AS ENUM ('ADMIN', 'GERENTE', 'COORDENADOR',
'PESQUISADOR', 'LOJISTA');
CREATE TYPE status_aprovacao AS ENUM ('PENDENTE', 'APROVADO', 'REJEITADO');
CREATE TYPE status_alocacao AS ENUM ('AGENDADA', 'EM_ANDAMENTO',
'CONCLUIDA');

CREATE TABLE usuarios (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    senha_hash VARCHAR(255) NOT NULL,
    perfil tipo_perfil NOT NULL,
    data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE marcas (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE modelos (
    id SERIAL PRIMARY KEY,
    marca_id INT REFERENCES marcas(id),
    nome VARCHAR(50) NOT NULL,
    categoria VARCHAR(30),
    ano_inicio_fabricacao INT
);

CREATE TABLE regioes (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    coordenador_id INT REFERENCES usuarios(id)
);

CREATE TABLE lojas (
    id SERIAL PRIMARY KEY,
    nome_fantasia VARCHAR(100) NOT NULL,
    endereco TEXT NOT NULL,
    regiao_id INT REFERENCES regioes(id),
    responsavel_cadastro_id INT REFERENCES usuarios(id),
    status status_aprovacao DEFAULT 'PENDENTE',
```

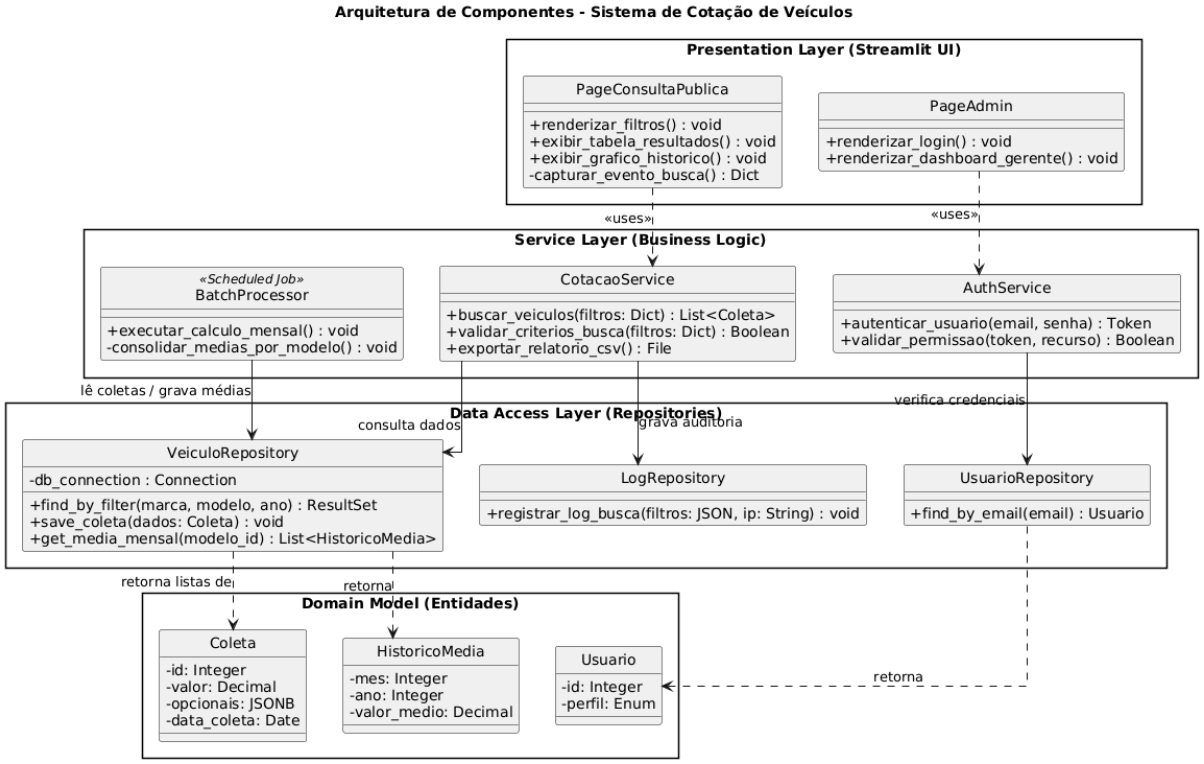
```
    data_cadastro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE alocacoes_pesquisa (
    id SERIAL PRIMARY KEY,
    coordenador_id INT REFERENCES usuarios(id),
    pesquisador_id INT REFERENCES usuarios(id),
    loja_id INT REFERENCES lojas(id),
    data_inicio_semana DATE NOT NULL,
    status status_alocacao DEFAULT 'AGENDADA'
);

CREATE TABLE coletas_veiculos (
    id SERIAL PRIMARY KEY,
    alocacao_id INT REFERENCES alocacoes_pesquisa(id),
    modelo_id INT REFERENCES modelos(id),
    preco_coletado DECIMAL(10, 2) NOT NULL,
    ano_modelo INT NOT NULL,
    ano_fabricacao INT NOT NULL,
    opcionais JSONB,
    observacoes TEXT,
    data_coleta TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE historico_medias_mensais (
    id SERIAL PRIMARY KEY,
    modelo_id INT REFERENCES modelos(id),
    mes INT NOT NULL,
    ano INT NOT NULL,
    media_preco DECIMAL(10, 2) NOT NULL,
    total_coletas INT NOT NULL,
    data_processamento TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(modelo_id, mes, ano)
);

CREATE TABLE logs_consultas_usuario (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    filtros_aplicados JSONB,
    ip_origem VARCHAR(45)
);
```

Minerva

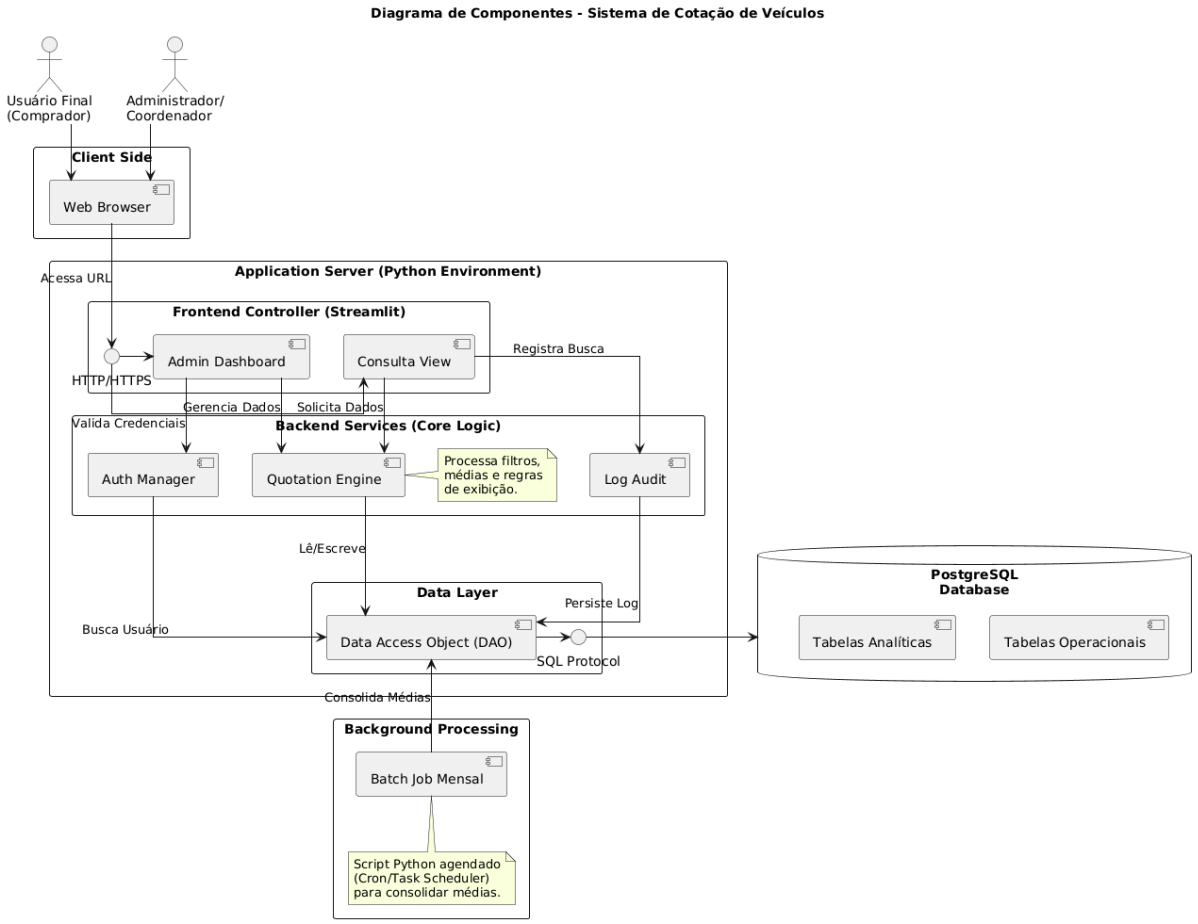
Arquitetura de componentes:

O sistema adota uma arquitetura em camadas (Layered Architecture), utilizando Python como linguagem core e PostgreSQL como sistema gerenciador de banco de dados relacional (RDBMS). A escolha tecnológica visa atender aos requisitos de consistência transacional (ACID) e flexibilidade na modelagem de dados semi-estruturados (Logs e Opcionais de Veículos).

- Diagrama de classe:



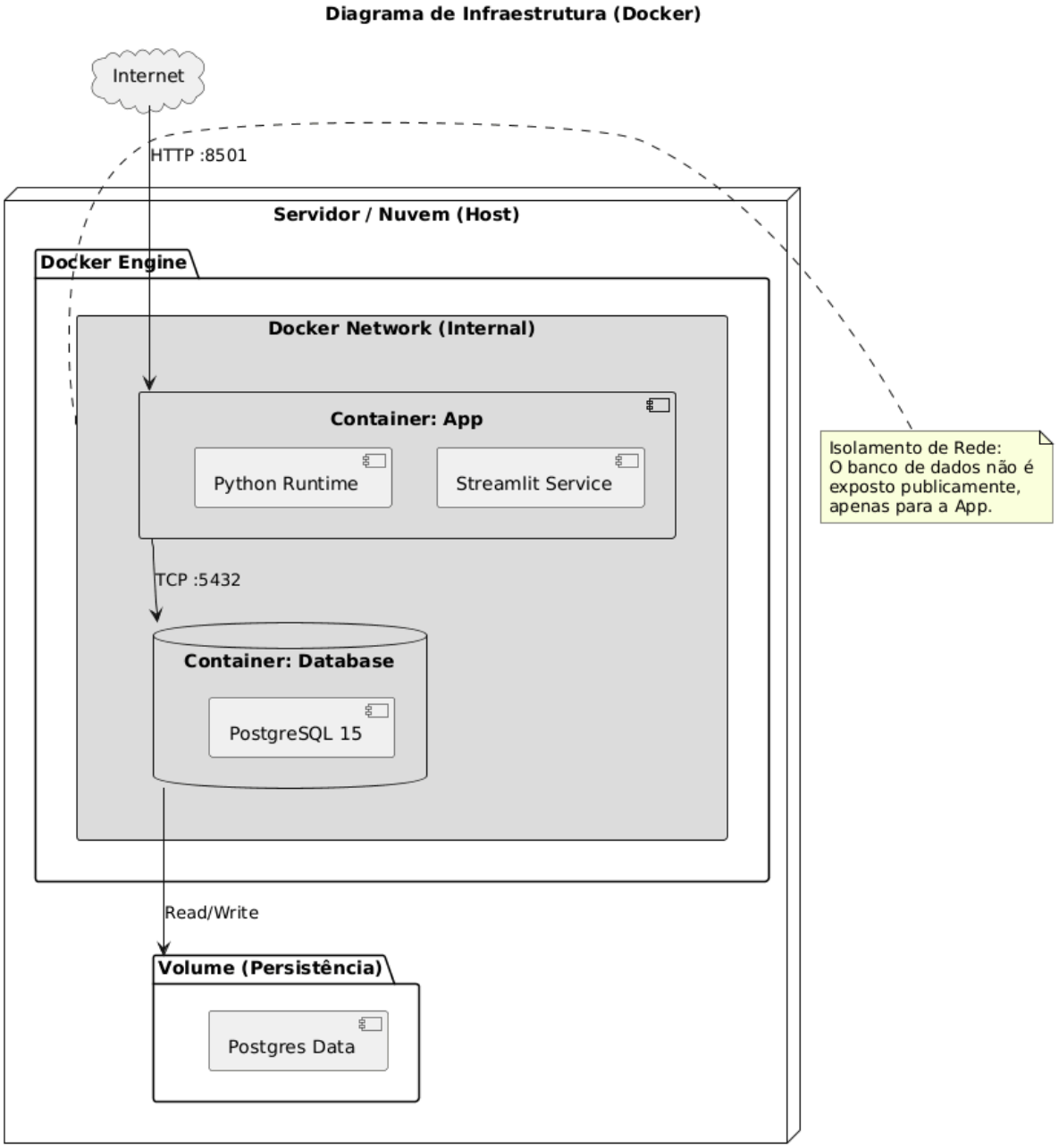
- Diagrama de componentes:



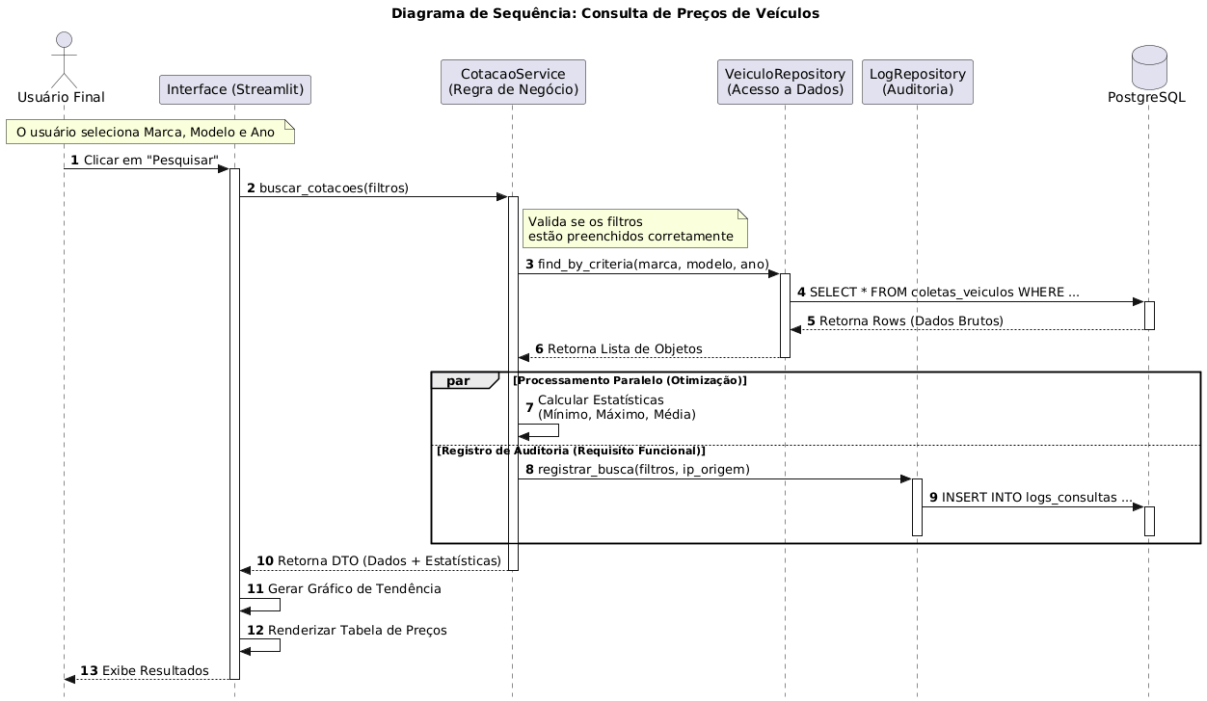
Infraestrutura e Containerização

A aplicação foi projetada para ser agnóstica de ambiente, utilizando **Docker** para orquestração. O sistema é composto por dois containers isolados (App e Banco) comunicando-se via rede interna virtual (Docker Network), garantindo portabilidade e fácil replicação do ambiente de desenvolvimento para produção.

- Diagrama de Implantação

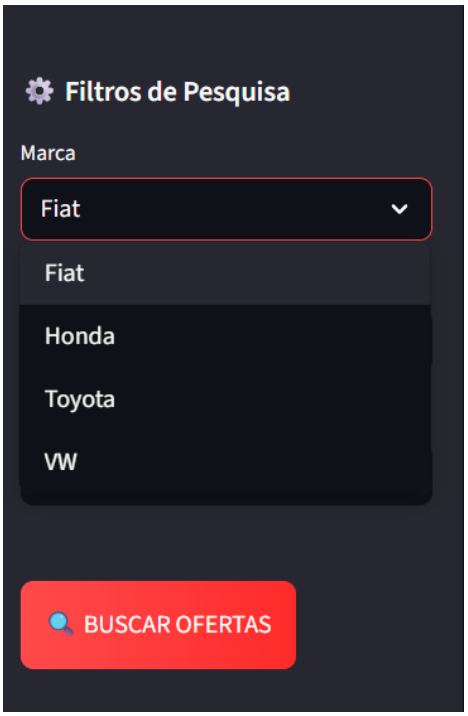


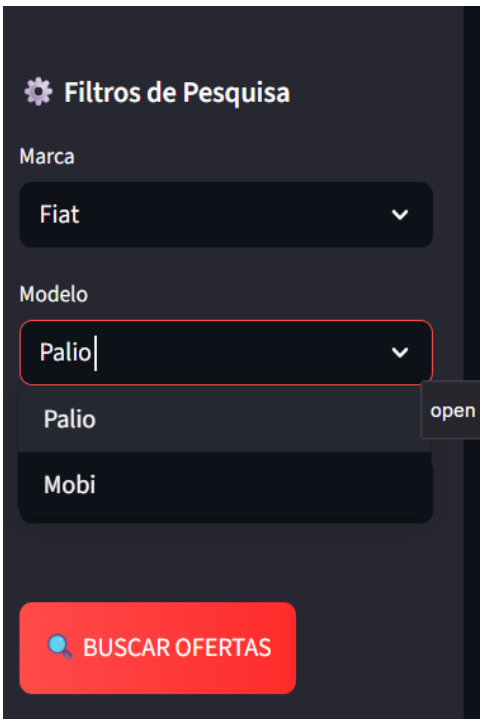
- Diagrama de Sequência:



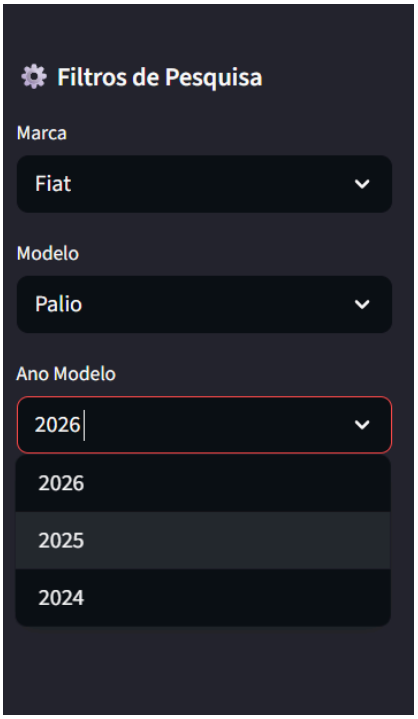
Minerva

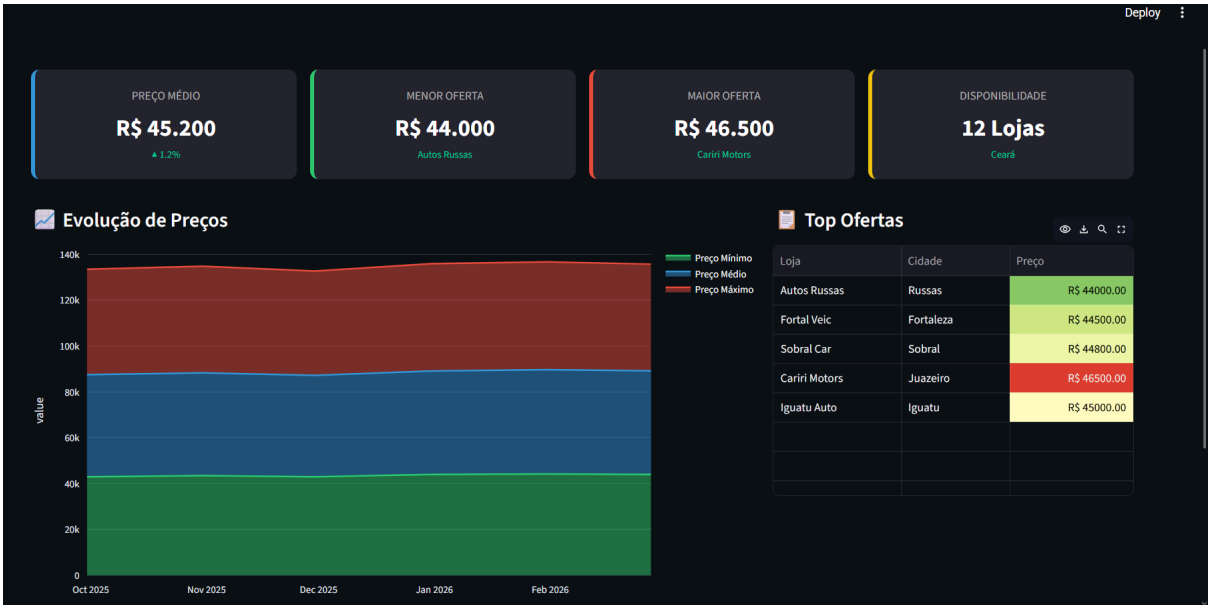
Protótipo - Interface de usuário.





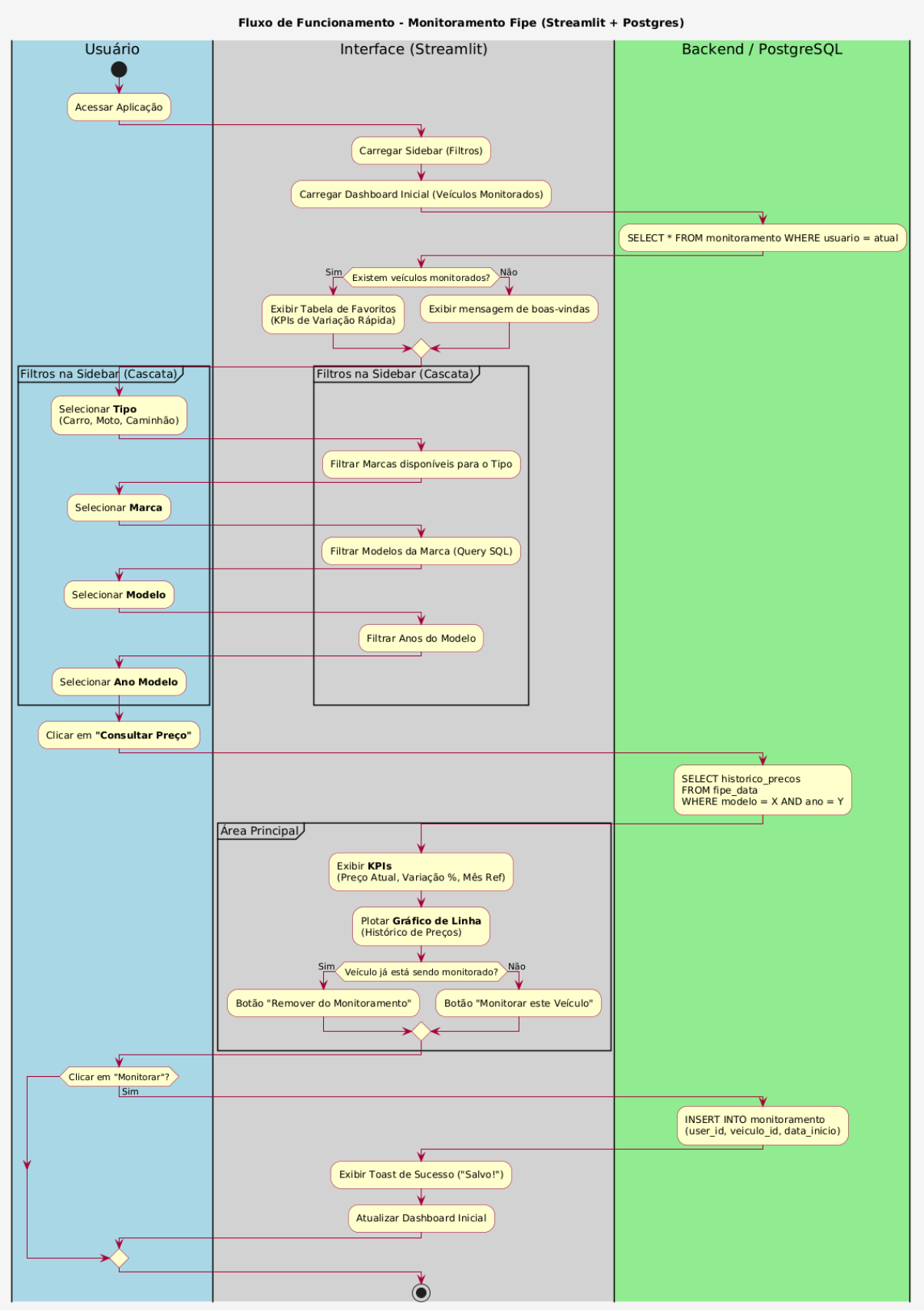
Minerva


Filtros de Pesquisa
Marca
Fiat
Modelo
Palio
Ano Modelo
2026
2026
2025
2024


Deploy
PREÇO MÉDIO
R$ 45.200
▲ 1.2%
MENOR OFERTA
R$ 44.000
Autos Russas
MAIOR OFERTA
R$ 46.500
Cariri Motors
DISPONIBILIDADE
12 Lojas
Ceará
Evolução de Preços
Preço Mínimo
Preço Médio
Preço Máximo
value
140k
120k
100k
80k
60k
40k
20k
0
Oct 2025
Nov 2025
Dec 2025
Jan 2026
Feb 2026
Top Ofertas
Loja
Cidade
Preço
Autos Russas
Russas
R$ 44000.00
Fortal Veic
Fortaleza
R$ 44500.00
Sobral Car
Sobral
R$ 44800.00
Cariri Motors
Juazeiro
R$ 46500.00
Iguatu Auto
Iguatu
R$ 45000.00

Minerva

Fluxo de funcionamento da interface.



Minerva

Technical Design (TD)

Esta seção detalha a implementação técnica do componente CotacaoService, responsável por orquestrar a comunicação entre a interface Streamlit e o banco de dados PostgreSQL.
A lógica de negócios será encapsulada na classe CotacaoService. Esta classe segue o padrão Singleton para conexão com o banco e Repository para abstração de queries.

```python
from typing import List, Dict, Optional
import pandas as pd
from dataclasses import dataclass

@dataclass
class FiltroBusca:
    marca_id: int
    modelo_id: int
    ano_modelo: Optional[int] = None

class CotacaoService:
    def __init__(self, db_connection):
        self.conn = db_connection

    def listar_opcoes_filtro(self) -> Dict[str, List]:

        pass

    def buscar_ofertas(self, filtros: FiltroBusca) -> pd.DataFrame:

        pass

    def calcular_kpis(self, df: pd.DataFrame) -> Dict:

        pass

    def obter_historico_preco(self, modelo_id: int) -> pd.DataFrame:

        pass

    def registrar_auditoria(self, filtros: FiltroBusca) -> None:

        pass
```

Onde:

listar_opcoes_filtro:

Retorna estrutura hierárquica para popular os selects: { 'Fiat': ['Palio', 'Mobi'], 'Honda': ['Civic'] } Cache Strategy: @st.cache_data (TTL 24h)

buscar_ofertas:

Executa a query principal. Retorna: DataFrame contendo [Loja, Cidade, Preço, Data, Opcionais]

calcular_kpis:

Processa o DataFrame para extrair métricas de negócio. Retorna: { 'media': float, 'min': float, 'max': float, 'total_ofertas': int }

obter_historico_preco:

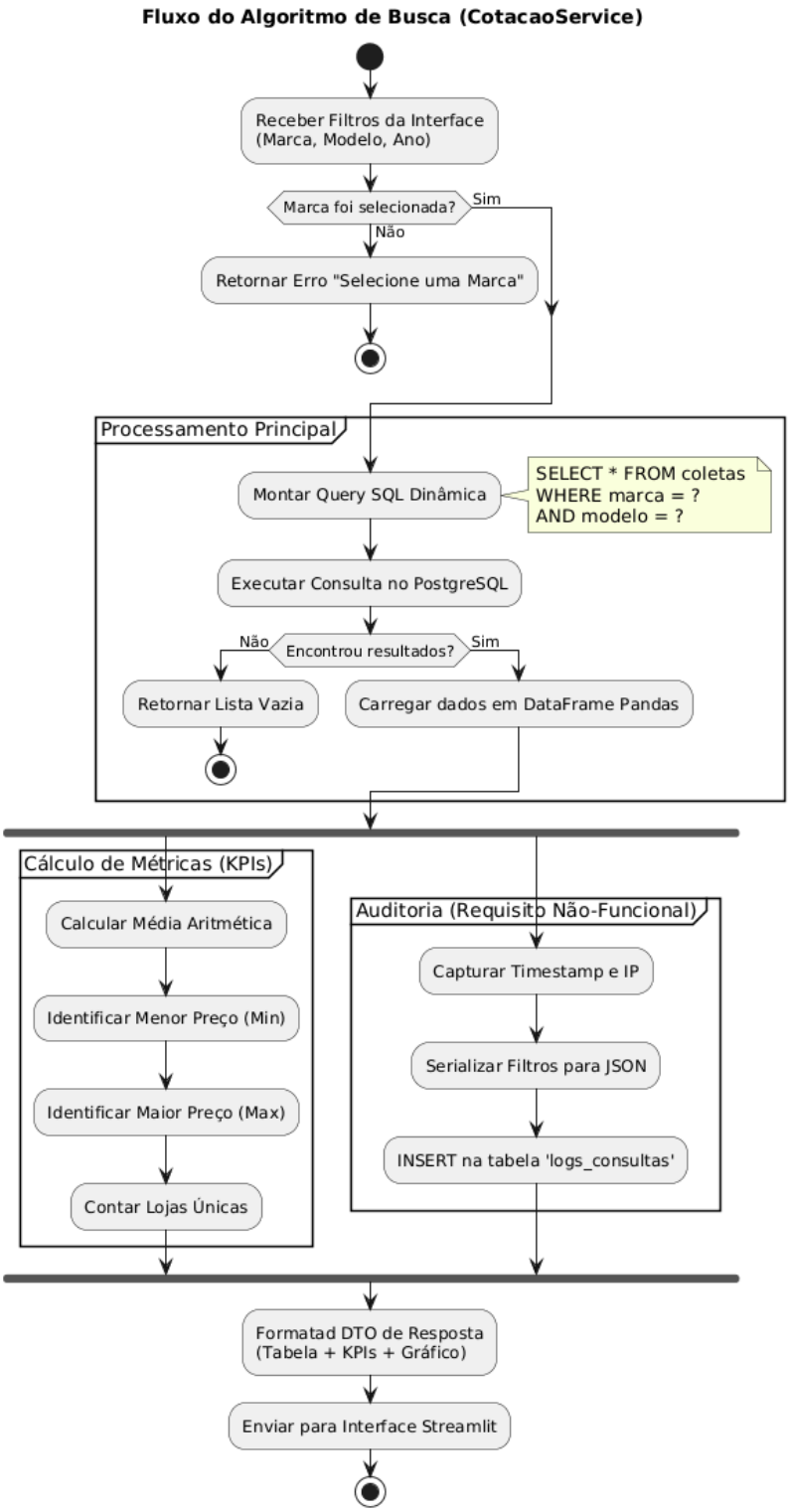
Busca a evolução de preços nos últimos 6 meses para o gráfico. Tabela fonte: historico_medias_mensais (Batch) ou agregação em tempo real.

registrar_auditoria:

Grava o log da pesquisa conforme requisito não-funcional de auditoria.

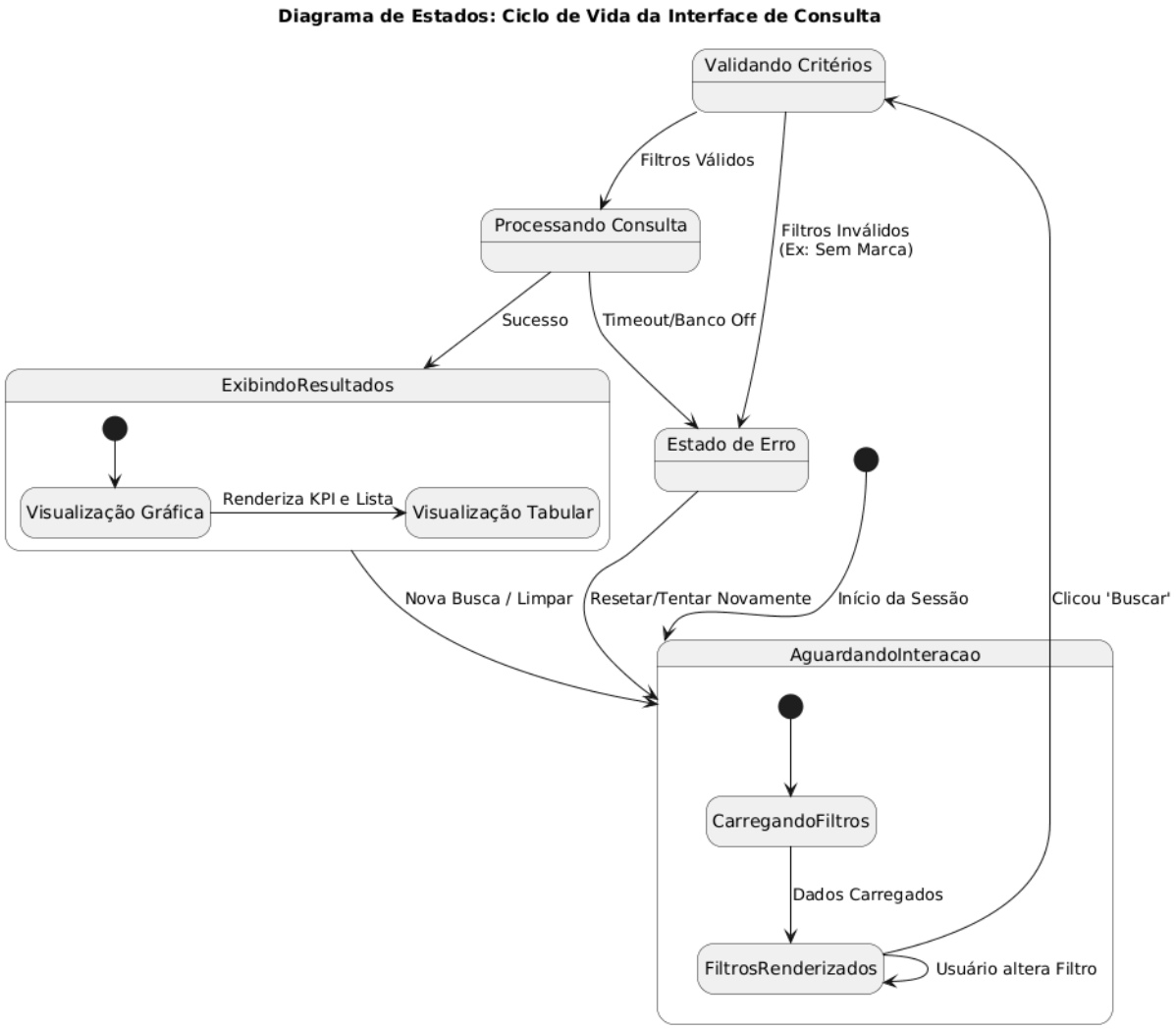
Minerva

Diagrama de Atividades:



Minerva

Diagrama de Máquina de Estados



Minerva

Core SQL

Esta é a query que alimentará o método buscar_ofertas. Ela realiza o JOIN entre 5 tabelas para trazer o dado completo para o usuário.

```sql
SELECT
    m.nome AS modelo,
    cv.ano_modelo,
    cv.preco_coletado AS preco,
    cv.data_coleta,
    cv.opcionais,            -- Campo JSONB
    l.nome_fantasia AS loja,
    l.endereco,
    r.nome AS regiao
FROM
    coletas_veiculos cv
JOIN
    modelos m ON cv.modelo_id = m.id
JOIN
    alocacoes_pesquisa ap ON cv.alocacao_id = ap.id
JOIN
    lojas l ON ap.loja_id = l.id
JOIN
    regioes r ON l.regiao_id = r.id
WHERE
    m.marca_id = :marca_id
    AND m.id = :modelo_id
    AND (:ano_modelo IS NULL OR cv.ano_modelo = :ano_modelo)
ORDER BY
    cv.preco_coletado ASC;
```

Estratégia de Indexação

Devido à complexidade da query principal de consulta, que envolve múltiplas junções (JOINs), foram projetados índices B-Tree para otimizar o tempo de resposta e reduzir o custo computacional do banco de dados.

CREATE INDEX idx_coletas_modelo ON coletas_veiculos(modelo_id);
CREATE INDEX idx_modelos_marca ON modelos(marca_id);
CREATE INDEX idx_coletas_data ON coletas_veiculos(data_coleta);

Minerva

Estrutura de dados de Respostas

O sistema utilizará a biblioteca Pandas para manipulação dos dados em memória, facilitando a integração com o Streamlit e Plotly.

Coluna	Tipo de Dado	Descrição
modelo	string	Nome do veículo (ex: "Palio 1.0")
preco	float	Valor monetário para cálculos
loja	string	Nome fantasia para exibição
regiao	string	Localização geográfica (Filtro implícito)
opcionais	object (dict)	Dados extraídos do JSONB (ex: Ar, Trava)
data_coleta	datetime	Para ordenação temporal

Minerva

Fluxo de Auditoria

Para atender ao requisito *"Sistema armazena dados relativos à consulta para análise"*, toda vez que o botão "Buscar" for acionado, o seguinte comando SQL será executado de forma assíncrona (ou sequencial, dada a baixa carga):

```sql
INSERT INTO logs_consultas_usuario (data_hora, filtros_aplicados, ip_origem)
VALUES (
    CURRENT_TIMESTAMP,
    :filtros_json,  -- Ex: {"marca": "Fiat", "modelo": "Palio", "ano": 2026}
    :user_ip
);
```

Plano de Testes e Aceitação

Casos de Teste Funcionais (Caixa Preta)

ID	Cenário	Passos de Execução (Action)	Resultado Esperado (Response)	Critério de Êxito
CT01	**Consulta com Sucesso (Caminho Feliz)**	1. Acessar a página inicial. 2. Selecionar Marca "Fiat". 3. Selecionar Modelo "Palio". 4. Clicar em "Pesquisar".	O sistema deve exibir: - Cards com Preço Médio, Mínimo e Máximo. - Gráfico de histórico. - Tabela com lista de lojas.	Dados exibidos correspondem aos registros ativos no banco.

CT02	**Validação de Campos Obrigatórios**	1. Acessar a página. 2. Não selecionar Marca ou Modelo. 3. Clicar em "Pesquisar"	O sistema deve exibir um alerta visual (Toast ou mensagem vermelha): "Por favor, selecione uma Marca e um Modelo".	A busca não é disparada no banco de dados.
CT03	**Busca Sem Resultados (Empty State)**	1. Selecionar uma combinação válida (ex: "Ferrari 2026"). 2. Clicar em "Pesquisar". 3. (Assumindo que não há coletas para este carro).	O sistema deve exibir mensagem amigável: "Nenhuma cotação encontrada para os filtros selecionados."	Interface não quebra (crash) e mostra feedback claro.
CT04	**Visualização de Histórico de Preços**	1. Realizar uma busca de sucesso (CT01). 2. Verificar o componente de gráfico.	O gráfico deve renderizar a evolução do preço médio nos últimos 6 meses (se houver dados históricos na tabela historico_medias_mensais).	Gráfico plotado corretamente via Plotly.

Teste Automatizado

```
import pytest
from services.cotacao_service import CotacaoService

def test_calculo_kpis_deve_retornar_media_correta():


    dados_mock = [
        {'preco': 100.00},
        {'preco': 200.00},
        {'preco': 300.00}
    ]

    service = CotacaoService()


    resultado = service.calcular_kpis(dados_mock)

    # 3. Assertion (Validação)
    assert resultado['media'] == 200.00
    assert resultado['min'] == 100.00
    assert resultado['max'] == 300.00
    assert resultado['total'] == 3
```

Relatório de Evolução

Este documento consolida o ciclo de vida completo do desenvolvimento do Sistema de Cotação de Veículos (Minerva Motors). O projeto partiu de uma concepção inicial focada em requisitos funcionais e evoluiu para uma arquitetura de software robusta, orientada a contêineres e performance.

1. Refinamento do Modelo de Dados: A modelagem inicial (Dia 1) previa apenas o armazenamento relacional padrão. Na entrega final, evoluímos para o uso de tipos semi-estruturados (JSONB) para flexibilizar o cadastro de opcionais e implementamos Índices B-Tree para otimizar a performance de consultas complexas.
2. Maturação Arquitetural: O desenho inicial de componentes foi expandido para incluir uma Camada de Serviço (Service Layer) e Repositórios, desacoplando a regra de negócio da interface gráfica.
3. Infraestrutura e Deploy: O projeto saiu de uma execução local (script Python simples) para uma arquitetura containerizada via Docker Compose, garantindo isolamento e reprodutibilidade do ambiente.
4. Robustez e Auditoria: Adicionamos diagramas comportamentais (Máquina de Estados) para tratar falhas de interface e implementamos o Log de Auditoria (Requisito Não-Funcional S3M-16), garantindo rastreabilidade das ações do usuário.
5. Para a camada de apresentação (Frontend), optou-se pela utilização do framework Streamlit. Essa escolha foi estratégica visando a otimização do tempo de desenvolvimento (Rapid Application Development). Como a lógica de negócios e o banco de dados foram priorizados nas etapas iniciais, o Streamlit permitiu uma integração ágil e desacoplada na fase final, eliminando a necessidade de desenvolver uma interface complexa em HTML/CSS/JS e permitindo foco total na validação dos requisitos funcionais e na inteligência de dados.