Documentação de Projeto

para o sistema

BusCars

Versão 2.0

Projeto de sistema elaborado pelo(s) aluno(s) Lucas Araujo Borges de Lima e Luis Gustavo Vaz e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor João Paulo Aramuni, orientação acadêmica do professor Cleiton Silva Tavares e orientação de TCC II do professor João Paulo Aramuni.

08/08/2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Entrega 1	07/09/25	Finalização do documento do projeto BusCars	1.0
Entrega 2	28/09/25	Finalização do documento do projeto BusCars	2.0

1. Introdução

Este documento de projeto do sistema **BusCars** serve como um alicerce técnico e estrutural para a elaboração e implementação de uma plataforma robusta e intuitiva de agregação e gerenciamento de veículos. A intenção primária deste documento é proporcionar uma visão detalhada dos modelos de domínio que fundamentam o sistema, assim como dos modelos de design que orientam sua construção. O documento é essencialmente técnico e destina-se a guiar a equipe de desenvolvimento via uma série de especificações meticulosamente organizadas, desenhos de arquitetura, e diretrizes de implementação em sincronia com os requisitos e objetivos delineados no documento de visão do sistema.

A funcionalidade e a inovação do sistema BusCars residem na sua capacidade de transformar e modernizar a maneira como a pesquisa e a compra de veículos são realizadas no mercado automotivo brasileiro. O sistema BusCars aborda como a plataforma irá melhorar a eficiência da busca, padronizar as informações de diferentes fontes e aprimorar a experiência do usuário por meio de uma interface amigável e de fácil utilização.

Por meio desta documentação, busca-se definir os padrões técnicos e os procedimentos necessários para garantir uma integração e escalabilidade eficazes, contemplando a aplicação de tecnologias avançadas para otimizar as operações e fortalecer a segurança do sistema. Este projeto foi desenvolvido para atender diretamente às necessidades e expectativas de **Lucas Amaral Paes Leme Maestro**, o *stakeholder* e gestor no setor automotivo, que deseja digitalizar e expandir a operação de compra e venda de veículos.

Lucas Amaral tem uma experiência significativa no setor automotivo na região metropolitana de Belo Horizonte. A digitalização da operação por meio da aplicação BusCars permitirá a otimização de seu fluxo de trabalho, que hoje exige a consulta manual de diversas plataformas, para uma gestão centralizada e mais eficiente. Com a implementação do sistema, o objetivo é tornar a curadoria dos anúncios de veículos mais eficazes, abrangendo desde a busca de dados em diferentes bases até a visualização e comparação, proporcionando uma experiência completa para os usuários finais.

Assim, a aplicação BusCars foi idealizada não apenas para modernizar a gestão interna do serviço, mas também para impulsionar o crescimento do negócio, proporcionando um sistema confiável, ágil e de fácil acesso tanto para compradores quanto para vendedores.

2. Modelos de Usuário e Requisitos

Nesta seção serão apresentadas as informações sobre os usuários e os requisitos do projeto. Além disso, serão apresentados o diagrama de caso de uso e as histórias de usuário.

2.1 Descrição de Atores

Os atores identificados para a utilização do sistema BusCars são os Clientes, os Vendedores, o ***Stakeholder*** e os Administradores. Os **Clientes** são os usuários finais que buscam veículos, utilizando a plataforma para centralizar suas pesquisas e obter informações detalhadas sobre as ofertas do mercado. **Vendedores** são os profissionais do setor que utilizam o sistema para acompanhar o mercado e, no futuro, poderão publicar seus próprios anúncios com maior riqueza de detalhes. O ***Stakeholder*** é o visionário do projeto, responsável por fornecer o conhecimento do mercado automotivo e garantir que o produto atenda às necessidades do setor. Por fim, os **Administradores** são os responsáveis pela gestão e operação do sistema, cuidando da integridade dos dados, monitorando as coletas e garantindo a eficiência da plataforma. Todos esses atores colaboram para que o BusCars seja uma solução completa e confiável no cenário de compra e venda de veículos.

2.2 Modelos de Usuários

Esta seção tem o propósito de descrever os modelos de usuários desenvolvidos por meio da implementação de personas. Com esse intuito, foram elaboradas personas representativas que funcionarão como orientações durante o processo de design e desenvolvimento da plataforma. Essas personas foram construídas com base em uma compreensão aprofundada das necessidades e características dos usuários-alvo. Na Tabela 1 é apresentada a persona do **Cliente**, representando o usuário que busca e compara veículos na plataforma. Na Tabela 2 é apresentada a persona do **Vendedor**, representando o profissional que acompanha o mercado e, futuramente, poderá publicar seus próprios anúncios. Já na Tabela 3 é apresentada a persona do **Administrador**, representando o usuário responsável por gerenciar e coordenar a base de dados do sistema.

Eduardo Pena	
Descrição	Ana Beatriz é uma profissional ocupada que vive em Belo Horizonte e está em busca de um veículo seminovo para uso diário. Ela valoriza seu tempo e busca uma forma mais eficiente e confiável de pesquisar carros, evitando ter que visitar múltiplos sites e lojas físicas sem uma pré-seleção. Ela é bastante conectada e utiliza a internet como sua principal ferramenta de pesquisa.
Dores	• A busca por veículos é fragmentada, exigindo que ela acesse vários *marketplaces* diferentes, o que é demorado e confuso. • A falta de padronização nas informações dificulta a comparação direta entre os anúncios. • Ela se sente insegura sobre a real condição e o preço justo de um veículo.
Objetivos	• Encontrar o carro ideal que se encaixe em seu orçamento e necessidades. • Tomar uma decisão de compra mais informada e segura, com acesso a dados comparativos.

	• Economizar tempo no processo de pesquisa, centralizando a busca em uma única plataforma confiável.
Tarefas	• Utilizar a barra de busca e os filtros avançados da plataforma. • Comparar diferentes ofertas de veículos. • Visualizar o preço de referência da Tabela FIPE para um anúncio. • Identificar anúncios duplicados para evitar perder tempo com ofertas repetidas. • Salvar buscas e veículos favoritos para revisão posterior.

Tabela 1. Persona Eduardo Pena

Henrique Mota	
Descrição	Henrique Mota é proprietário de uma pequena revenda de carros seminovos em Belo Horizonte. Ele tem um estoque limitado e, por isso, busca otimizar a visibilidade de seus veículos para atrair clientes qualificados. Atualmente, ele utiliza vários *marketplaces*, mas percebe que as plataformas existentes não valorizam o diferencial de seu estoque.
Dores	• Falta de visibilidade e destaque para seus veículos em meio a um grande volume de anúncios genéricos. • O gerenciamento de anúncios em diferentes plataformas é demorado e ineficiente. • A dificuldade em atrair *leads* qualificados e em lidar com contatos de baixo valor ou curiosos, que não resultam em vendas concretas.
Objetivos	• Aumentar a visibilidade de seu estoque para atrair clientes mais qualificados. • Simplificar o gerenciamento de seus anúncios em um único lugar. • Utilizar uma plataforma que valorize a qualidade de seus veículos e permita a criação de anúncios mais completos e atrativos.
Tarefas	• Acompanhar os preços de mercado na plataforma para precificar seu estoque de forma competitiva. • Buscar veículos para compra em revendas. • No futuro, criar e gerenciar anúncios próprios, adicionando descrições detalhadas e fotos de alta resolução.

Tabela 2. Persona Henrique Mota

Caio Silva	
Descrição	Caio Silva é o responsável pela gestão e operação da plataforma BusCars. Ele atua nos bastidores, garantindo que o sistema funcione de forma fluida, confiável e segura. Sua principal função é monitorar a integridade das informações e manter a saúde técnica do produto. Ele garante a qualidade e consistência do BusCars, assegurando que a experiência do usuário seja fluida apesar da complexa integração de dados.
Dores	• Falta de visibilidade sobre falhas de integração (*web scraping* e *APIs*). • Inconsistência nos dados agregados, que podem não estar padronizados ou desatualizados. • A identificação manual de anúncios duplicados ou fraudulentos consome muito tempo e recursos.
Objetivos	• Garantir a operação contínua e a alta disponibilidade do sistema. • Assegurar a qualidade e a consistência dos dados apresentados aos usuários. • Otimizar o processo de moderação e manutenção da base de dados
Tarefas	• Monitorar o log de erros das integrações com outras plataformas. • Validar novos dados e remover anúncios inconsistentes ou duplicados. • Acompanhar métricas de desempenho do sistema e do banco de dados. • Realizar atualizações e manutenções na plataforma.

Tabela 3. Persona Caio Silva

2.3 Modelo de Casos de Uso e Histórias de Usuários

Nesta subseção são apresentados o diagrama de casos de uso do sistema e as histórias dos usuários, que serão apresentadas nas seções 2.3.1 e 2.3.2 respectivamente.

2.3.1 Diagrama de Caso de Uso

A Figura 1 representa o diagrama de casos de uso referentes ao sistema proposto. No diagrama é possível ver 3 atores principais, Cliente, Vendedor, e Administrador, que são usuários do sistema.

A busca é a principal funcionalidade do BusCars, o ponto de entrada para a experiência de todos os usuários, sejam eles Clientes ou Vendedores. Diferente dos *marketplaces* tradicionais, que apresentam dados de forma isolada, o BusCars consolida informações de diversas fontes para oferecer uma pesquisa centralizada e inteligente.

Para o Cliente, a busca é a ferramenta fundamental para encontrar o carro ideal. A plataforma permite que ele pesquise utilizando filtros avançados, como marca, modelo, ano e localização, e acesse em uma única tela os anúncios unificados de diferentes sites. Isso não apenas poupa tempo, mas também oferece uma visão comparativa do mercado, com dados enriquecidos como o valor da Tabela FIPE.

Já para o Vendedor, a busca serve como uma poderosa ferramenta de inteligência de mercado. Ao utilizar os mesmos filtros, ele pode analisar o comportamento dos preços de mercado para precificar seu próprio estoque de forma mais competitiva. Além disso, a busca por oportunidades de compra permite que ele encontre veículos com potencial de revenda, otimizando seu negócio.

O Administrador opera nos bastidores do BusCars garantindo a integridade e a saúde da plataforma. Sua função principal é monitorar a coleta de dados para assegurar que os anúncios sejam importados sem falhas. Ele também valida o conteúdo, removendo informações inconsistentes ou fraudulentas, e identifica duplicatas para manter a base de dados limpa. Para tomar decisões estratégicas, ele utiliza a visualização de métricas do sistema, analisando o desempenho e o comportamento dos usuários.

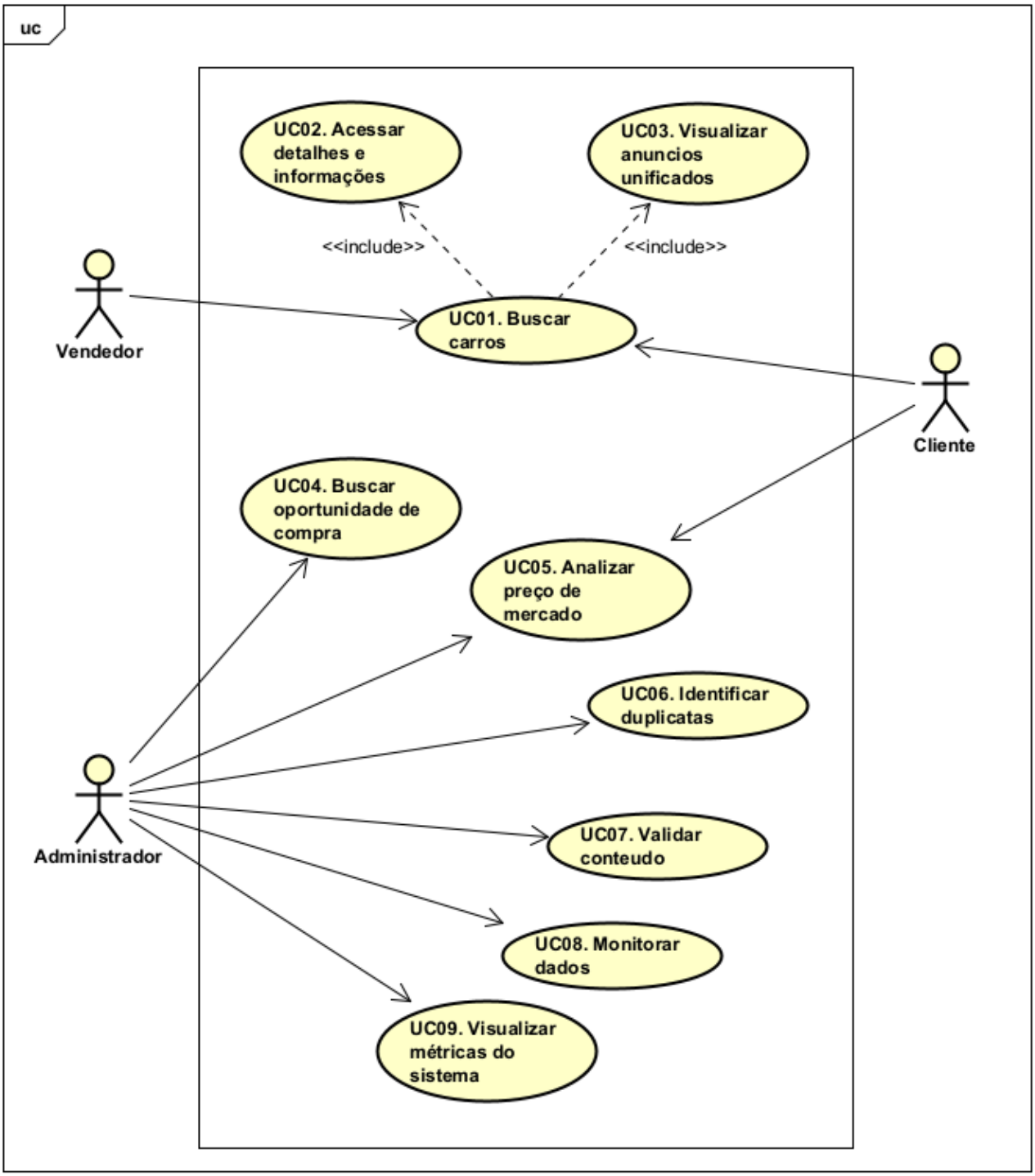

Figura 1. Diagrama de Caso de Uso

2.3.2 Historias de Usuário

Nesta seção são listadas as histórias de usuários elicitadas para o sistema do BusCars. Para fins de organização, utilizam-se identificadores no formato US#ID, em que US refere-se a *User Story*. Assim, as histórias de usuário identificadas para o sistema são:

US01. Como Vendedor, desejo buscar carros na plataforma, para encontrar veículos com potencial de revenda e analisar o mercado.

US02. Como Vendedor, desejo poder acessar os detalhes e as informações de um veículo, para ter os dados completos e tomar uma decisão de compra mais informada.

US03. Como Vendedor, desejo visualizar todos os anúncios de um veículo em uma única página, para comparar as ofertas de forma rápida e eficiente.

US04. Como Cliente, desejo buscar veículos na plataforma, para encontrar o carro ideal que se encaixe nas minhas necessidades.

US05. Como Cliente, desejo poder acessar os detalhes e as informações de um veículo, para ter uma visão completa antes de tomar uma decisão.

US06. Como Cliente, desejo visualizar os anúncios de um veículo em uma única página, para evitar ter que abrir várias abas no navegador e poupar tempo.

US07. Como Cliente, desejo analisar o preço de um carro em relação ao mercado, para saber se a oferta é justa.

US08. Como Administrador, preciso analisar o preço médio de um veículo, para manter a base de dados da plataforma com informações precisas.

US09. Como Administrador, preciso identificar anúncios duplicados na base de dados, para garantir a qualidade e a unicidade das informações.

US10. Como Administrador, preciso validar o conteúdo dos anúncios, para remover dados inconsistentes e garantir a credibilidade da plataforma.

US11. Como Administrador, desejo monitorar a coleta de dados de cada fonte, para garantir que o sistema esteja sempre atualizado.

US12. Como Administrador, desejo visualizar as métricas de desempenho do sistema, para entender o comportamento dos usuários e guiar futuras melhorias.

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Nesta seção, são apresentados o Diagrama de Sequência do Sistema (DSS) e o Contrato de Operações associados aos principais fluxos de negócio do BusCars. O objetivo destes diagramas é descrever as interações de alto nível entre os atores externos (Cliente, Vendedor e Administrador) e o sistema. A Figura XX representa o conjunto de diagramas que modela os 12 casos de uso do sistema, desde a busca de veículos até a gestão da curadoria. O DSS e seus contratos relacionados

garantem a formalização dos requisitos funcionais, estabelecendo as pré-condições necessárias e as pós-condições garantidas para o sucesso de cada operação.

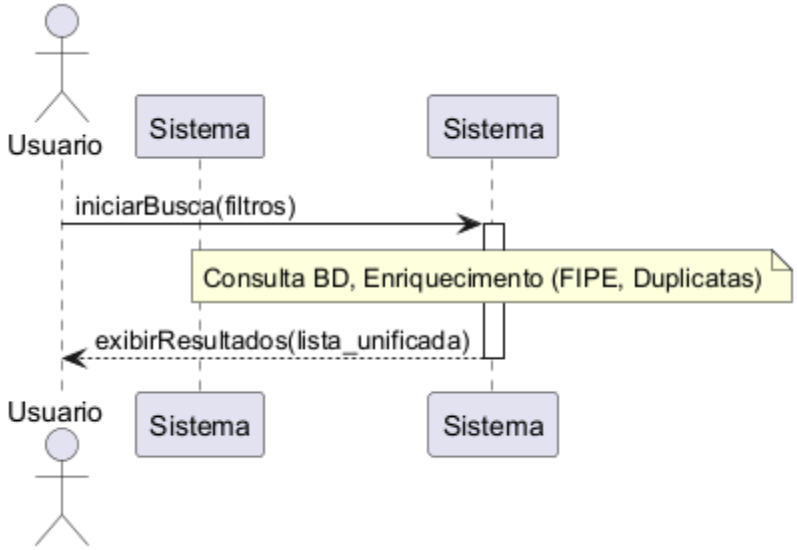

Figura 2. DSS: Busca de Veículos

Contrato	Operação
Operação	iniciarBusca(filtros: Mapa<String, String>)
Referências cruzadas	US01, US04
Pré-condições	O sistema deve conter anúncios normalizados no BD.
Pós-condições	1. O sistema retorna uma lista de Anuncio que atendem aos filtros. 2. A lista é processada pelo Serviço de Enriquecimento.

Tabela 4. DSS: Busca de Veículos

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Busca de Veículo Padrão. A Figura 2 ilustra o diagrama, onde o usuário envia os filtros de pesquisa e o sistema processa os resultados através do Serviço de Enriquecimento. Este diagrama se relaciona aos casos de uso US01 e US04, assegurando que os resultados sejam sempre unificados e enriquecidos antes de serem exibidos, cumprindo o valor central da plataforma.

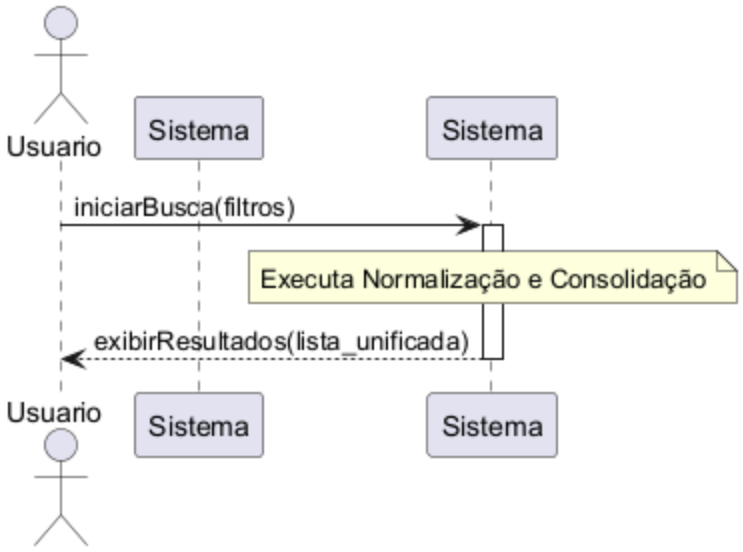

Figura 3. DSS: Visualização Unificada

Contrato	Operação
Operação	retornarAnunciosUnificados()
Referências cruzadas	US03, US06
Pré-condições	A busca inicial deve ter retornado resultados válidos.
Pós-condições	1. Os dados de todos os anúncios são normalizados e padronizados para exibição comparativa.

Tabela 5. DSS: Visualização Unificada

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Visualização Unificada. A Figura 3 ilustra o diagrama, onde o sistema executa a rotina de Normalização e Consolidação no *backend*. Este diagrama se relaciona aos casos de uso US03 e US06, garantindo que a Pós-condição de padronização seja executada no *backend*, cumprindo a promessa de visualização unificada e poupando tempo do usuário.

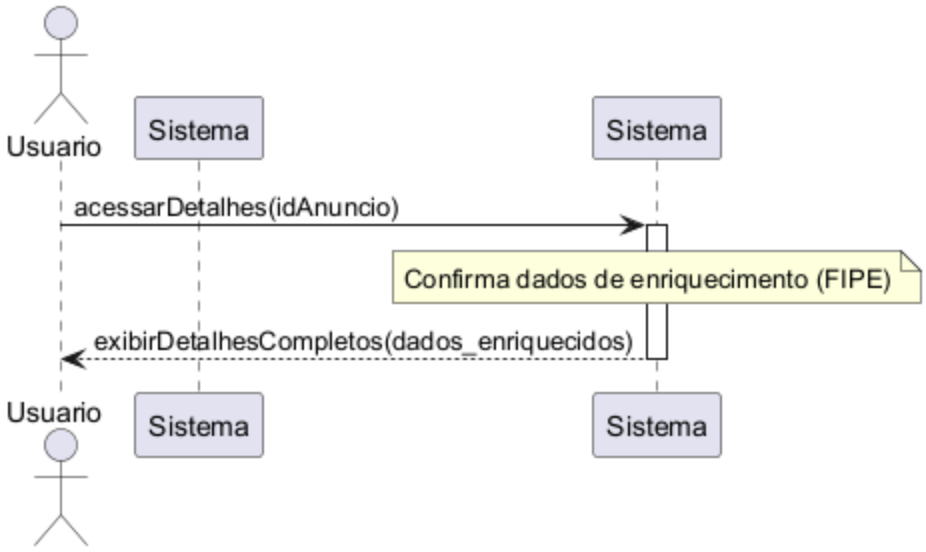

Figura 4. DSS: Detalhes do Veículo

Contrato	Operação
Operação	acessarDetalhes(idAnuncio)
Referências cruzadas	US02, US05
Pré-condições	O idAnuncio é válido e o anúncio está com *status* "Ativo".
Pós-condições	1. O sistema retorna todos os atributos e mídias do Anuncio e do Veiculo. 2. Os dados de enriquecimento (FIPE) são confirmados e exibidos.

Tabela 6. DSS: Detalhes do Veículo

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Acesso a Detalhes. A Figura 4 ilustra o diagrama, onde o usuário solicita o idAnuncio e o sistema realiza uma consulta completa. Este diagrama se relaciona aos casos de uso US02 e US05, assegurando que a Pós-condição de retorno inclua os dados de enriquecimento essenciais para que o Vendedor e o Cliente tomem decisões informadas.

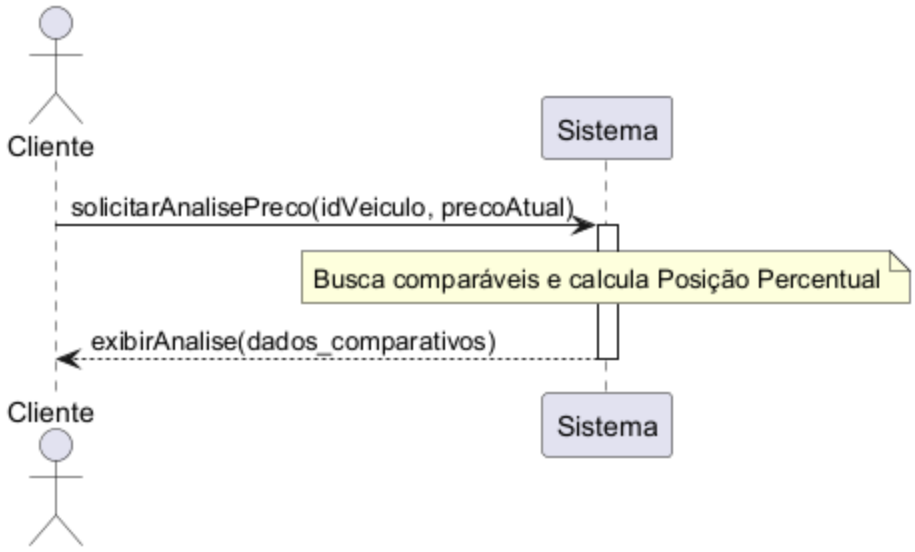

Figura 5. DSS: Análise de Preço do Cliente

Contrato	Operação
Operação	analisarPrecoMercado(idVeiculo, precoAtual)
Referências cruzadas	US07
Pré-condições	O sistema deve ter *dataset* de comparação suficiente (mínimo de 10 anúncios similares).
Pós-condições	1. O sistema calcula o preço médio e a posição percentual do precoAtual contra o mercado.

Tabela 7. DSS: Análise de Preço do Cliente

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Análise de Preço do Cliente. A Figura 5 ilustra o diagrama, onde o Cliente solicita a comparação de um veículo. Este diagrama se relaciona ao caso de uso US07, assegurando que o contrato garanta que o sistema execute o Cálculo da Inteligência para suportar a decisão do Cliente sobre o valor justo do veículo.

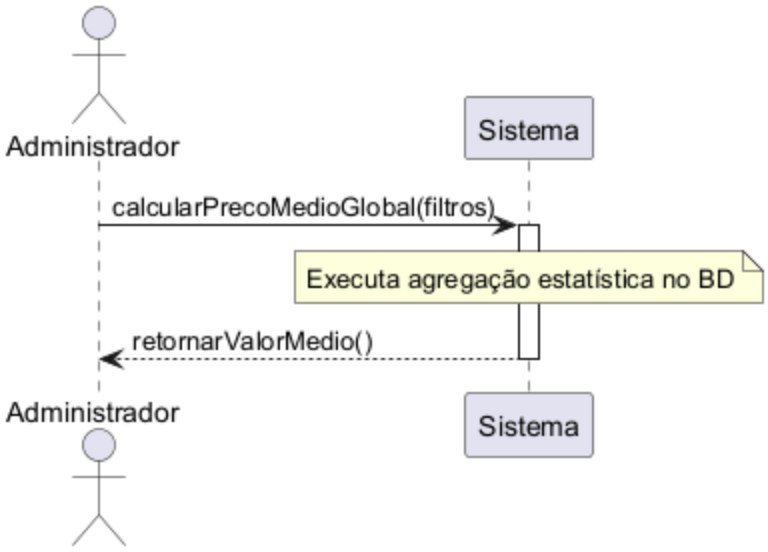

Figura 6. DSS: Média de Preço

Contrato	Operação
Operação	calcularPrecoMedioGlobal(filtros)
Referências cruzadas	US08
Pré-condições	O Administrador deve estar autenticado.
Pós-condições	1. Uma agregação estatística é executada no BD. 2. O valor médio é registrado e utilizado para calibrar o Serviço de Enriquecimento.

Tabela 8. DSS: Média de Preço

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Análise Média de Preço (Gestão). A Figura 6 ilustra o diagrama, onde o Administrador solicita a calibração de preços. Este diagrama se relaciona ao caso de uso US08, definindo que o contrato estabelece esta operação como uma função de calibração do sistema, essencial para manter a precisão do Serviço de Enriquecimento.

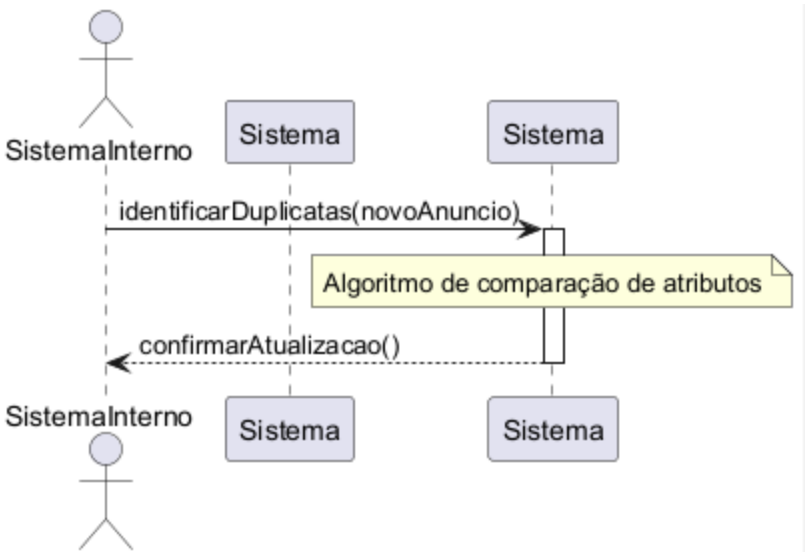

Figura 7. DSS: Identificar Duplicatas

Contrato	Operação
Operação	identificarDuplicatas(novoAnuncio)
Referências cruzadas	US09
Pré-condições	Um novo Anuncio deve ter sido coletado e persistido.
Pós-condições	1. O sistema executa um algoritmo de comparação contra o banco de dados. 2. O Anuncio e seus pares são atualizados, definindo isDuplicado como *TRUE*.

Tabela 9. DSS: Identificar Duplicatas

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Identificação de Duplicatas. A Figura 7 ilustra o diagrama, onde a interação é disparada pelo Sistema Interno. Este diagrama se relaciona ao caso de uso US09, garantindo que a Pós-condição de manutenção de dados seja executada pelo algoritmo de comparação, assegurando a unicidade e a qualidade do *dataset*.

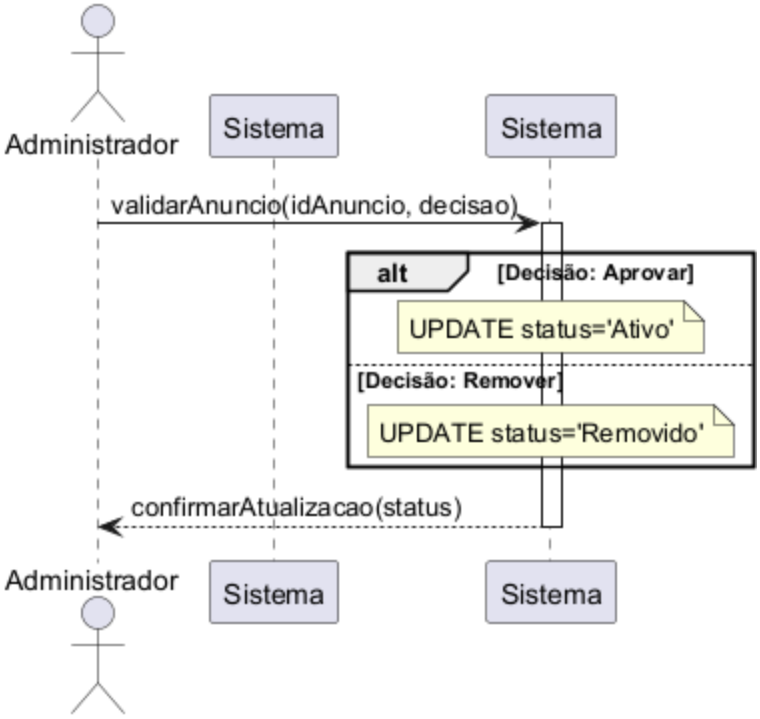

Figura 8. DSS: Validar Conteúdo

Contrato	Operação
Operação	validarAnuncio(idAnuncio, decisao)
Referências cruzadas	US10
Pré-condições	O Administrador deve estar autenticado. O anúncio está no *status* "Pendente" ou "Sinalizado".
Pós-condições	1. O *status* do Anuncio é alterado para "Ativo" (se aprovado) ou "Removido" (se rejeitado). 2. Se "Ativo", o anúncio se torna visível nas buscas.

Tabela 10. DSS: Validar Conteúdo

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Validação de Conteúdo. A Figura 8 ilustra o diagrama, onde o Administrador toma a decisão de moderação. Este diagrama se relaciona ao caso de uso US10, garantindo que a Pós-condição de atualização do *status* defina a visibilidade do anúncio, cumprindo o requisito de credibilidade da plataforma.

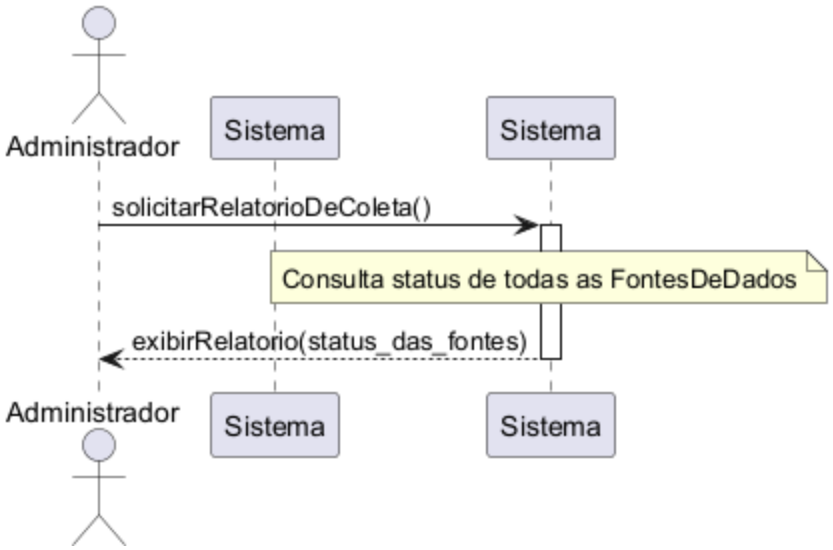

Figura 9. DSS: Coleta de Dados

Contrato	Operação
Operação	solicitarRelatorioDeColeta()
Referências cruzadas	US11
Pré-condições	O Administrador deve estar autenticado. A rotina de coleta deve ter sido executada recentemente.
Pós-condições	1. O sistema consulta o statusColeta de cada FonteDeDados. 2. Um RelatorioDeColeta é gerado e exibido na interface de administração.

Tabela 11. DSS: Coleta de Dados

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Monitoramento da Coleta de Dados. A Figura 9 ilustra o diagrama, onde o Administrador solicita o relatório de saúde. Este diagrama se relaciona ao caso de uso US11, garantindo que o contrato retorne um relatório que é um insumo para a Ação Corretiva, detalhando o *status* de saúde de cada fonte externa.

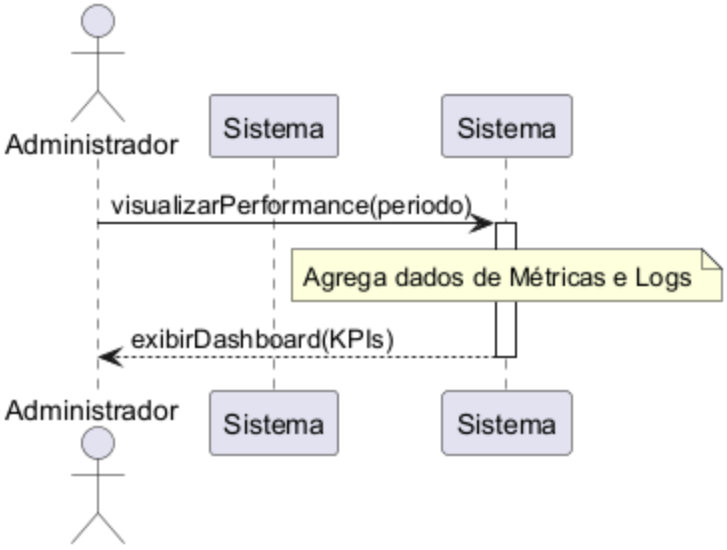

Figura 10. DSS: Métricas de Desempenho

Contrato	Operação
Operação	visualizarPerformance(periodo)
Referências cruzadas	US12
Pré-condições	O Administrador deve estar autenticado. O sistema deve ter dados registrados na classe Métrica.
Pós-condições	1. O sistema agrega e calcula dados de logs e eventos para o periodo solicitado. 2. Um *dashboard* visual (KPIs) é gerado e exibido na interface de administração.

Tabela 12. DSS: Métricas de Desempenho

Nesta seção, são exibidos o DSS e o Contrato de Operações do fluxo de Visualização de Métricas de Desempenho. A Figura 10 ilustra o diagrama, onde o Administrador solicita o *dashboard*. Este diagrama se relaciona ao caso de uso US12, garantindo que o contrato faça o Processamento dos dados de *logs* e eventos para gerar um *dashboard* estratégico.

3. Modelos de Projeto

Aqui, são apresentados os modelos de projeto do sistema proposto, que incluem o diagrama de Classes, detalhado na Seção 3.1, os diagramas de sequência e de comunicação, fornecidos na seção 3.2 e 3.3, respectivamente. Além disso, discutimos a arquitetura lógica, abordada na Seção 3.4, e os diagramas de estado, explorados na Seção 3.5. Por fim, são apresentados os diagramas de componentes e de implantação, descritos na Seção 3.6. Esses elementos constituem a base do projeto do sistema, fornecendo uma compreensão abrangente de sua estrutura e funcionamento.

3.1 Diagrama de Classes

Na Figura 11, encontra-se representado o diagrama de classes do sistema BusCars, exibindo todas as classes gerenciadas por ele. As classes correspondem aos componentes centrais, como Cliente, Vendedor, Anúncio e Veículo. Algumas dessas classes possuem estruturas compostas e, portanto, estão relacionadas a classes como a ScrapeDTO, que atua como um elemento-chave na obtenção de dados de outras plataformas. Essas classes representam as informações manipuladas pelo sistema durante a execução das operações. Todas as classes são persistidas no banco de dados, seguindo uma organização específica, detalhada na Seção 5. O diagrama ilustra como o sistema trata cada uma das classes, seus relacionamentos e composições. Essa estrutura é fundamental para o desenvolvimento das interfaces de usuário e para a lógica de negócios do sistema, facilitando a manipulação e interação com os dados. As funcionalidades da aplicação são construídas com base na definição e agregação dessas classes, possibilitando a prestação eficiente dos serviços oferecidos. O sistema BusCars, conforme ilustrado no diagrama de classes na Figura 11, é projetado para gerenciar eficientemente a agregação de veículos, cobrindo todo o processo desde a coleta de dados de diversas bases até a apresentação unificada dos anúncios. O fluxo operacional do sistema é orientado pela interação entre as classes principais: Cliente, Vendedor, Administrador, Anúncio, Veículo e ScrapeDTO.

O fluxo operacional do sistema começa com a coleta de dados de fontes externas, como iCarros, WebMotors e OLX. Utilizando a classe ScrapeDTO, o sistema consome e processa informações de anúncios disponíveis nessas plataformas. Uma vez que o sistema coleta os dados, as classes Anúncio e Veículo são usadas para formatar e persistir as informações, permitindo que a plataforma padronize como o conteúdo é apresentado. Essa padronização é essencial para que os usuários possam realizar buscas e comparações de forma eficiente, um processo que antes era impossível devido à fragmentação de dados. Durante o processo de coleta, o sistema permite que o administrador monitore o *status* e o desempenho das rotinas de extração de dados.

O Cliente (comprador) tem a opção de pesquisar por veículos utilizando a classe Cliente, que acessa os dados por meio de filtros e visualizações unificadas. Ao selecionar um veículo, ele pode acessar as informações completas e enriquecidas, como o valor da Tabela FIPE. Da mesma forma, o Vendedor utiliza suas funcionalidades para analisar os preços de mercado e buscar oportunidades de compra para seu estoque. Essa interação bidirecional garante que o sistema ofereça valor para ambos os perfis de usuários. O Administrador é o responsável por garantir a qualidade da informação. A classe Administrador permite que ele valide o conteúdo, monitore os dados e identifique e remova duplicatas, assegurando que os dados apresentados aos usuários sejam confiáveis e relevantes.

O sistema BusCars foi idealizado para modernizar a gestão interna dos negócios de seus usuários, mas também para impulsionar o crescimento do mercado de compra e venda de veículos. Ao centralizar dados e oferecer ferramentas de análise, a aplicação proporciona um sistema confiável, ágil e de fácil acesso, tanto para compradores quanto para vendedores.

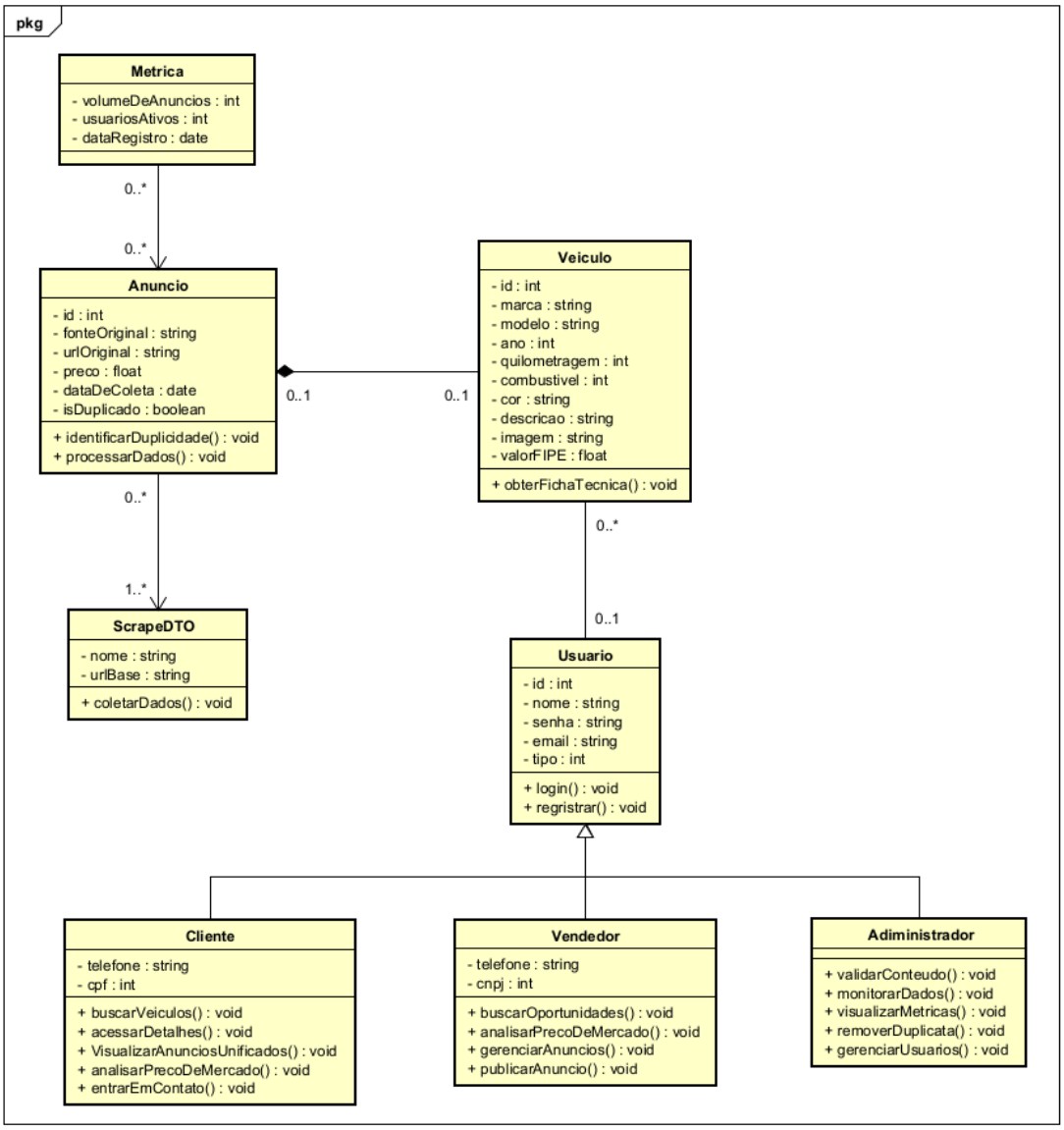

Figura 11. Diagrama de Classes

3.2 Diagramas de Sequência

Esta seção apresenta os Diagramas de Sequência, que modelam o sistema BusCars a ser desenvolvido. A finalidade é traçar os principais caminhos percorridos pelos usuários (Clientes, Vendedores e Administradores) durante a utilização da aplicação. Para isso, detalhamos as interações entre os diferentes componentes do sistema, incluindo as mensagens trocadas, e a ordem temporal necessária para o funcionamento adequado e eficiente das funcionalidades do BusCars.

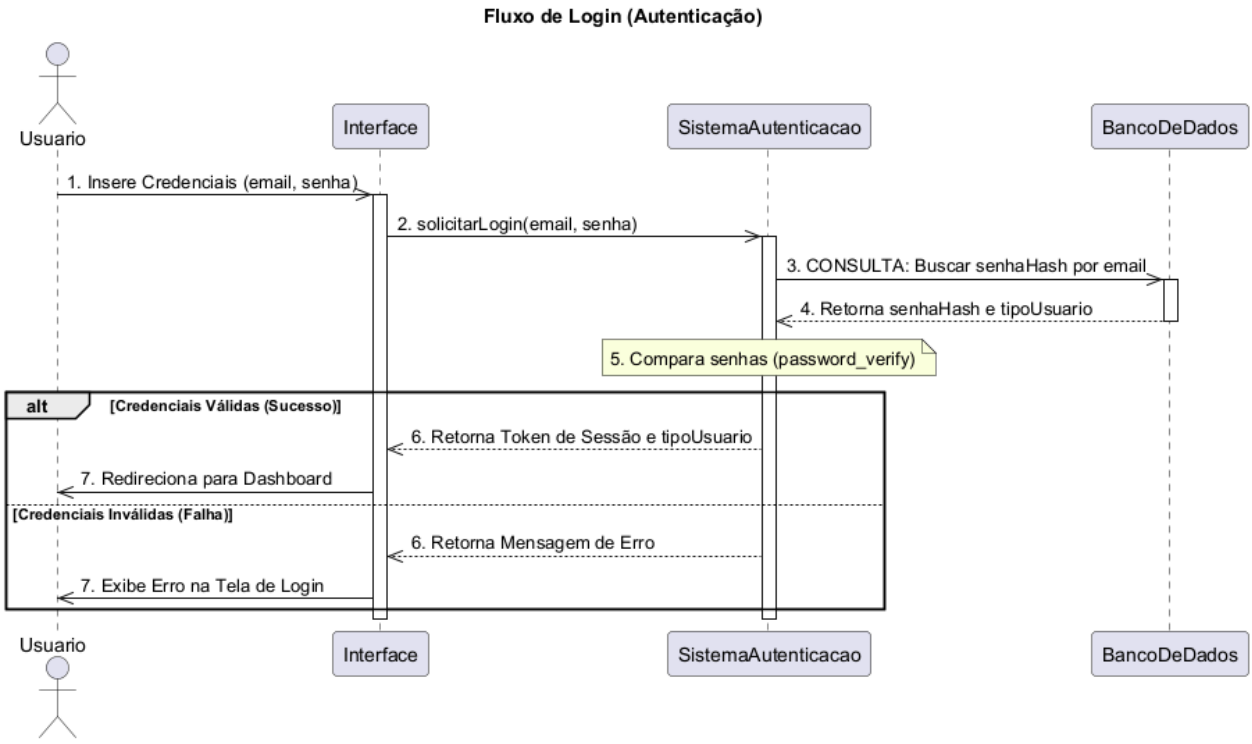

Figura 12. Diagrama de Sequência de login

A Figura 12 representa o fluxo de Login na plataforma BusCars, um processo estruturado para ser seguro e eficiente. O fluxo começa com a Iniciação, onde o Usuário insere seu *e-mail* e senha na interface de *login*. Em seguida, o Sistema de Autenticação executa a Validação, consultando o Banco de Dados para obter o *hash* da senha e compará-lo com a informação fornecida. A senha original jamais é exposta. O processo culmina no Resultado: em caso de Sucesso, o sistema gera um *token* de sessão e informa o tipoUsuario (Cliente, Vendedor ou Administrador) à interface, redirecionando o usuário para a área apropriada. Em caso de Falha, o sistema retorna uma mensagem de erro genérica ("Credenciais inválidas"), mantendo a segurança do acesso.

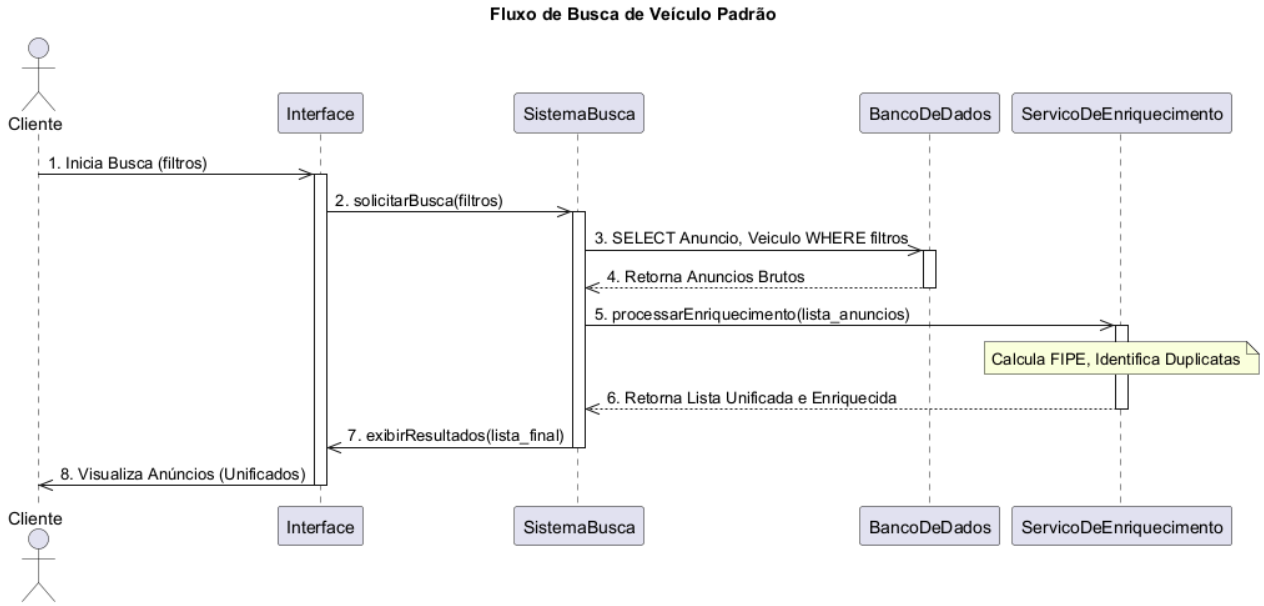

Figura 13. Diagrama de Sequência de busca com filtro

A Figura 13 representa o fluxo de Busca de Veículo Padrão do sistema BusCars. O Cliente (Comprador ou Vendedor) inicia o processo inserindo filtros de busca na Interface do usuário. Esta, por sua vez, envia a solicitação ao Sistema Busca, que consulta o Banco de Dados para recuperar os anúncios e veículos correspondentes. Após a consulta, o Sistema Busca aciona o Serviço de Dados (Enriquecimento), que processa a lista de resultados em tempo real para calcular o valor da Tabela FIPE e identificar duplicatas. O sistema recebe a lista final, formatada e processada, permitindo que o cliente visualize os anúncios unificados e enriquecidos, facilitando a tomada de decisão.

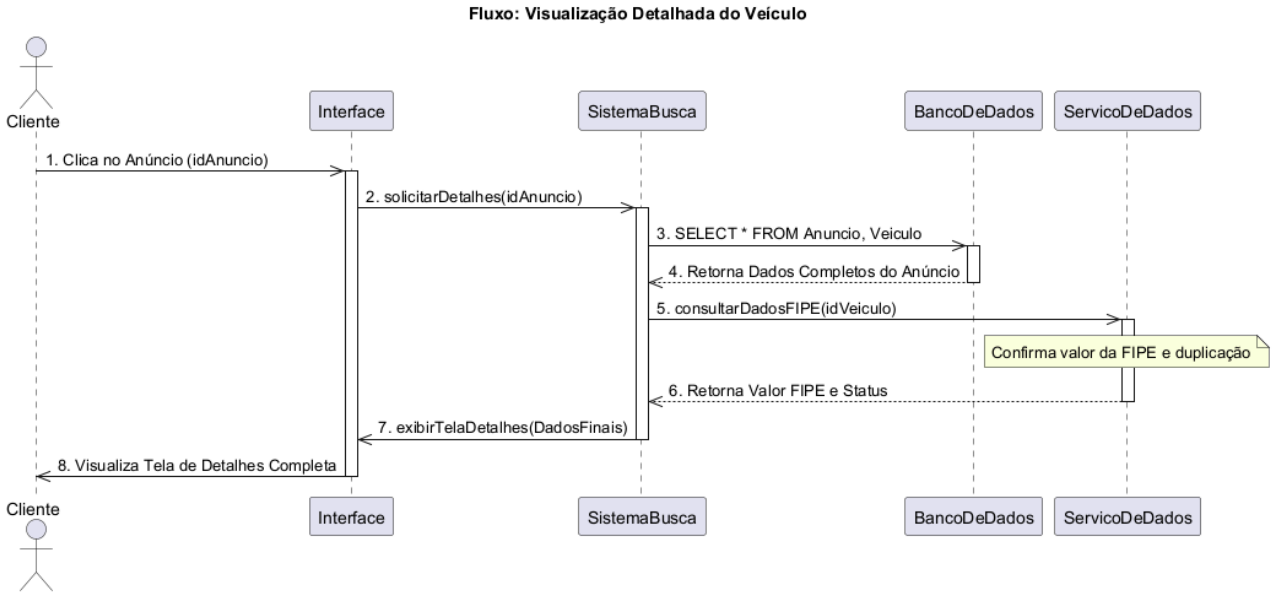

Figura 14. Diagrama de Sequência de visualização detalhada do veículo

A Figura 14 representa o fluxo de Visualização Detalhada do Veículo no BusCars, uma etapa crítica para a tomada de decisão do usuário. A jornada se inicia quando o Cliente ou Vendedor clica em um resultado de busca, enviando o idAnuncio para o *backend*. Em seguida, o Sistema de Busca realiza uma Consulta Completa no Banco de Dados para recuperar todos os atributos e mídias do anúncio e do veículo. Para garantir a precisão da análise, o Sistema de Busca aciona o Serviço de Dados (Enriquecimento), que confirma o valor da Tabela FIPE e o *status* de duplicação. Por fim, o Sistema de Busca consolida a descrição original, as fotos e todas as informações enriquecidas, enviando-as para a Interface, que exibe a Tela de Detalhes Completa ao usuário.

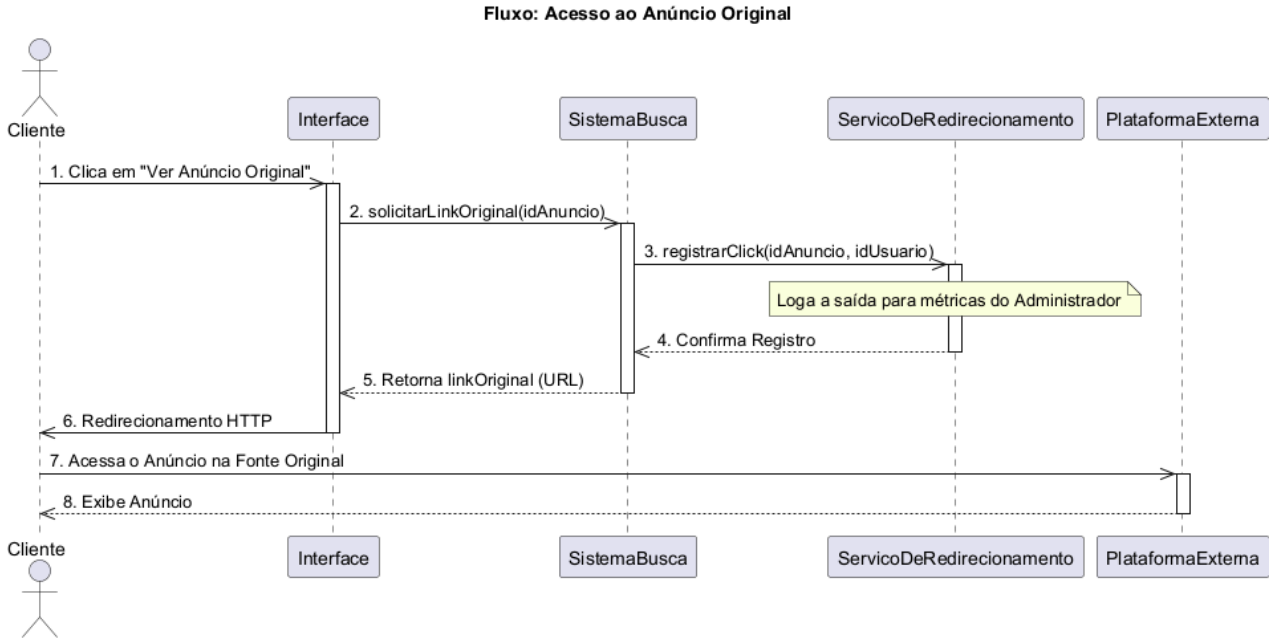

Figura 15. Diagrama de Sequência de acesso ao anúncio original

A Figura 15 representa o fluxo de Acesso ao Anúncio Original, um processo crucial para o modelo de negócio do BusCars. A sequência inicia quando o Cliente clica no botão para sair, enviando o ID do anúncio ao *backend*. Em seguida, o Sistema de Busca aciona o Serviço de Redirecionamento, que não só recupera o linkOriginal do anúncio, mas também registra o clique para fins de Registro Analítico do Administrador. Após esse registro, o *backend* retorna a *URL* original para a Interface do BusCars, que executa o Redirecionamento *HTTP* no navegador. O fluxo se conclui quando o navegador do usuário acessa a Plataforma Externa (como WebMotors ou iCarros), e o anúncio original é carregado.

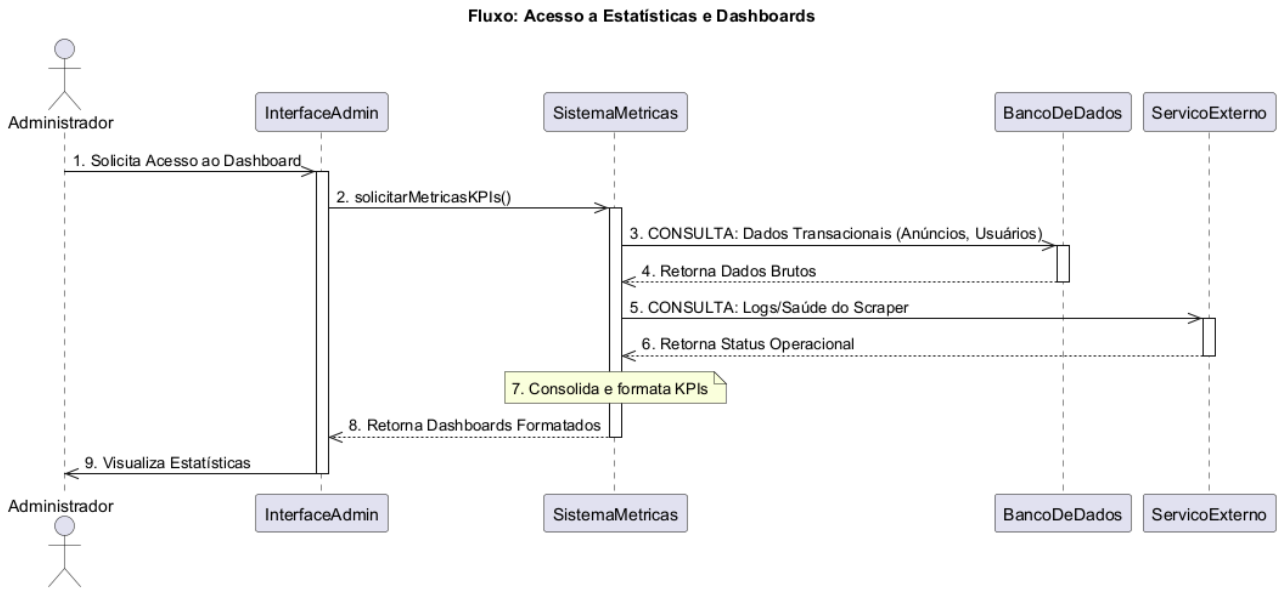

Figura 16. Diagrama de Sequência de acesso a estatísticas e dashboards

A Figura 16 representa o fluxo de Acesso a Estatísticas e Dashboards no BusCars, uma função essencial para o Administrador. A sequência começa com a Iniciação, onde o Administrador acessa a interface de gestão e solicita o dashboard de métricas. Em seguida, o Sistema de Métricas executa a Agregação de Dados, atuando como um orquestrador ao buscar informações em duas fontes primárias: o Banco de Dados, para dados de volume e transacionais, e os Serviços de Monitoramento Externo, para logs de saúde do sistema e performance. Na etapa de Processamento, o Sistema de Métricas consolida as informações brutas em KPIs (Indicadores-Chave de Desempenho), preparando-as para a análise. O fluxo é finalizado com a Apresentação, onde o painel formatado é exibido na Interface Admin, permitindo ao administrador tomar decisões estratégicas informadas.

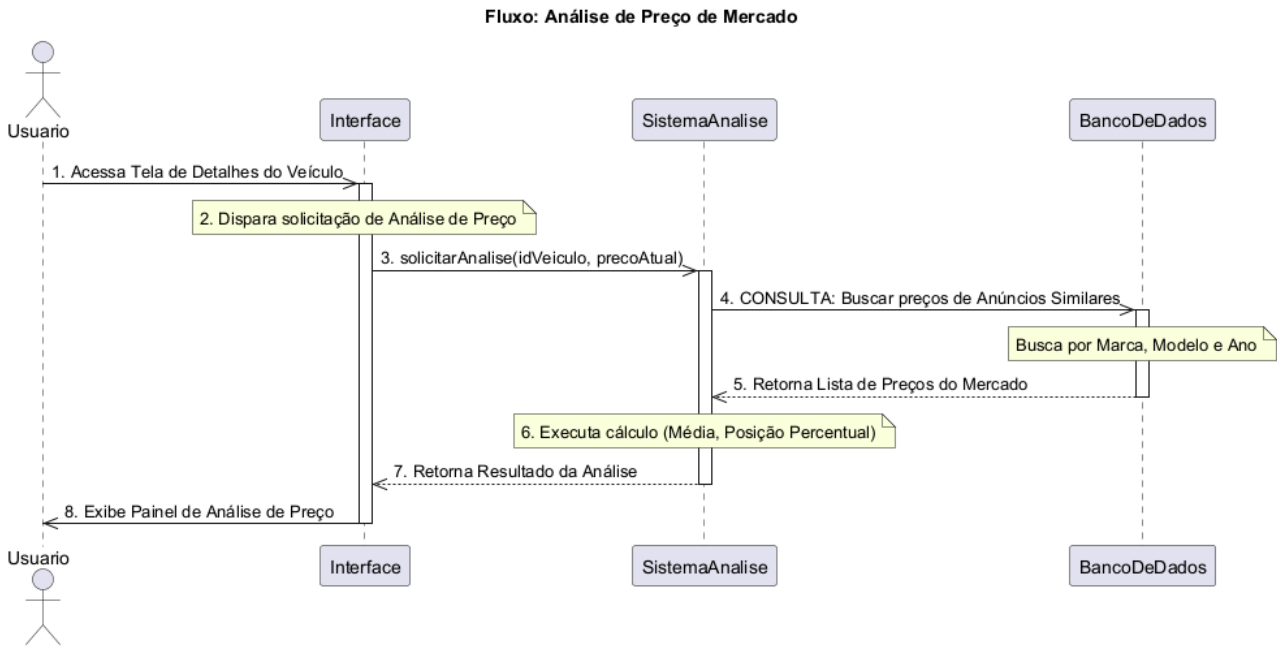

Figura 17. Diagrama de Sequência de análise de preço de mercado

A Figura 17 representa o fluxo de Análise de Preço de Mercado no BusCars, que transforma dados brutos em inteligência de mercado para o usuário. A Iniciação da solicitação ocorre automaticamente ao carregar a página de detalhes ou via um clique do Usuário (Vendedor ou Cliente). Em seguida, o Sistema de Análise executa a Consulta de Comparáveis, utilizando o idVeiculo para buscar no Banco de Dados todos os Anúncios similares que servem como *dataset* de comparação de mercado. Após receber a lista de preços, o sistema procede ao Cálculo da Inteligência, determinando o preço médio e a posição percentual do veículo em análise. O fluxo é finalizado com a Apresentação do Resultado, onde a Interface exibe um painel de fácil compreensão que indica se o preço do carro está justo, caro ou representa uma oportunidade.

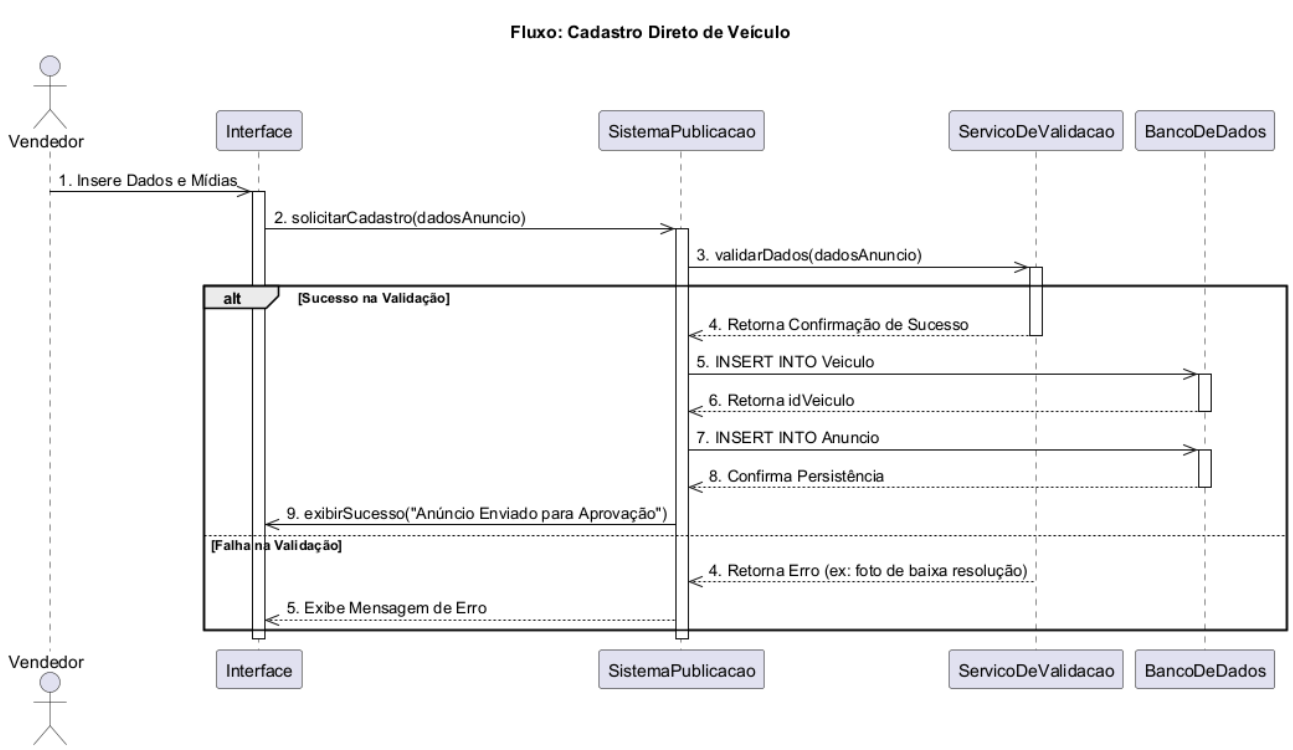

Figura 18. Diagrama de Sequência de cadastro direto de veículo

A Figura 18 representa o fluxo de Cadastro Direto de Veículo no BusCars, uma funcionalidade futura projetada com forte ênfase na curadoria de conteúdo. O processo se inicia com o Vendedor preenchendo um formulário detalhado, que exige informações mais completas que o padrão do mercado. Em seguida, o Sistema de Publicação aciona o Serviço de Validação, que realiza a Validação de Qualidade (verificando fotos e campos técnicos). Caso a validação falhe, o processo retorna um erro imediato ao vendedor. Se a validação for bem-sucedida, o sistema procede com a Persistência de Dados, inserindo primeiro o registro na tabela Veículo e, logo após, criando o registro do Anúncio, vinculando-o ao veículo. O fluxo é concluído com a notificação de sucesso ao Vendedor, indicando que o anúncio está pronto para a etapa de moderação pelo Administrador antes da publicação final.

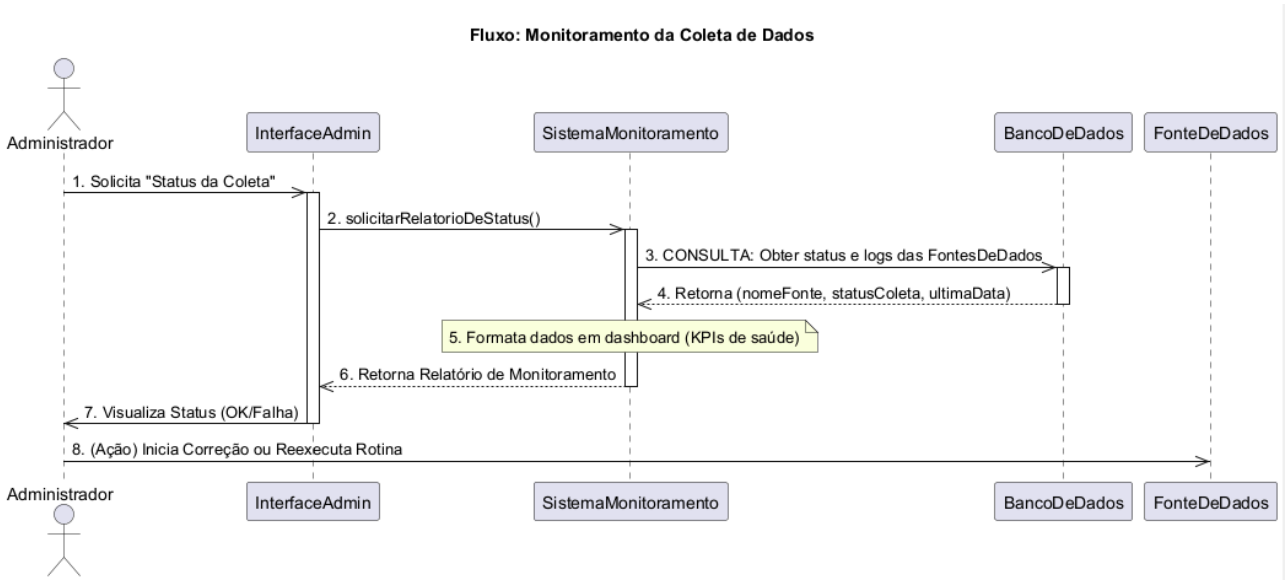

Figura 19. Diagrama de Sequência de monitoramento e coleta de dados

A Figura 19 representa o fluxo de Monitoramento da Coleta de Dados no BusCars, um processo essencial para a confiabilidade da plataforma como agregador. A sequência inicia com a Iniciação, onde o Administrador acessa uma seção específica da Interface Admin para solicitar o relatório de saúde. Em seguida, o Sistema de Monitoramento executa a Consulta de *Status* no Banco de Dados, extraindo o estado e o histórico de execução de todas as entradas da tabela Fonte de Dados. O sistema realiza a Análise e Formatação, convertendo os dados brutos em um relatório legível que indica quais fontes tiveram sucesso ou apresentaram falha. Por fim, o fluxo é concluído com a Ação Corretiva, onde o Administrador visualiza o relatório e pode tomar medidas imediatas, como reajustar a rotina de *scraping* para garantir que o sistema permaneça sempre atualizado.

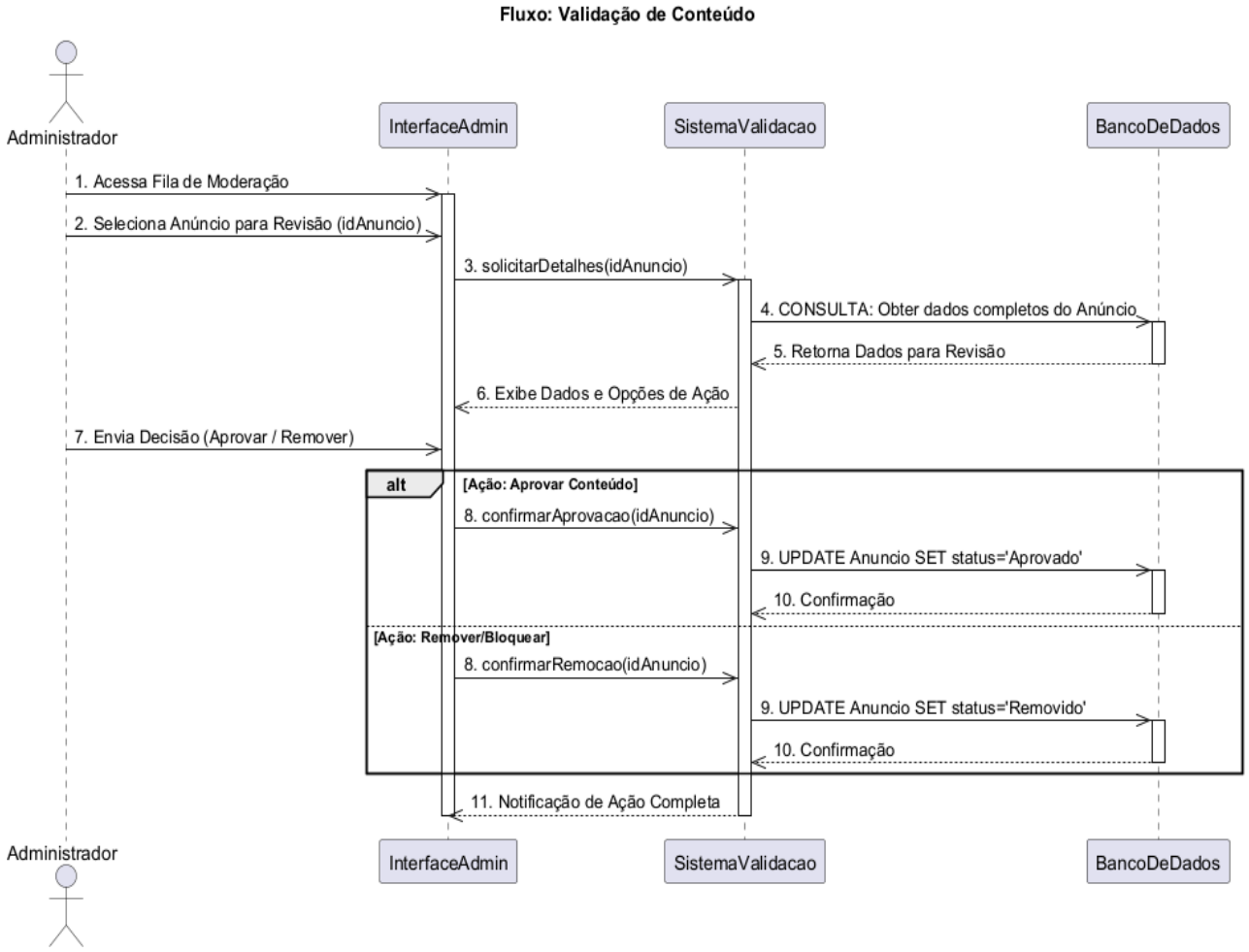

Figura 20. Diagrama de Sequência de validação de conteúdo

A Figura 20 representa o fluxo de Validação de Conteúdo no BusCars, que reflete o papel do Administrador como curador da plataforma. O processo começa com a Revisão de Conteúdo, onde o Administrador acessa a Interface Admin para analisar os detalhes do anúncio, buscando informações cruciais de segurança e consistência. Em seguida, o Sistema de Validação executa uma Consulta no Banco de Dados para buscar os dados completos do Anúncio e do Veículo, fornecendo o contexto total para a Tomada de Decisão. Após a análise humana, o Administrador envia a Ação final (Aprovar ou Remover) para o *backend*. O fluxo é concluído com a Atualização da Base, onde o Sistema de Validação altera o *status* do Anúncio no Banco de Dados, garantindo que apenas conteúdos aprovados sejam visíveis aos Clientes e Vendedores.

3.3 Diagramas de Comunicação

Nesta seção, são apresentados os Diagramas de Comunicação que modelam as trocas de mensagens dos principais Casos de Uso do sistema BusCars. Esses diagramas representam as interações estruturais e sequenciais entre os componentes da arquitetura para realizar uma

determinada funcionalidade, como a busca de veículos ou a validação de conteúdo. Dessa forma, os principais fluxos da aplicação são documentados, servindo como um mapa detalhado das comunicações que os compõem.

A Figura 21 representa o diagrama de comunicação responsável pelo fluxo de Busca, Visualização e Análise de Veículos. Nesse fluxo, o Usuário (Cliente ou Vendedor) inicia a comunicação por meio da Interface do Usuário (UI), com o objetivo de buscar veículos (UC01), acessar seus detalhes (UC02) e realizar a Análise de Preço de Mercado (UC07). Essa comunicação é recebida pelo *Controller*, que valida os filtros e define a ação. Após a validação, o *Controller* se comunica com o *Service*, que processa a lógica de negócios, interage com o *Repository* e o Banco de Dados para consultar os anúncios. O *Service* aciona o componente de Enriquecimento para calcular a FIPE e as médias antes de retornar a lista de resultados unificados para exibição. Esse diagrama contempla os casos de uso UC01, UC02 e UC07.

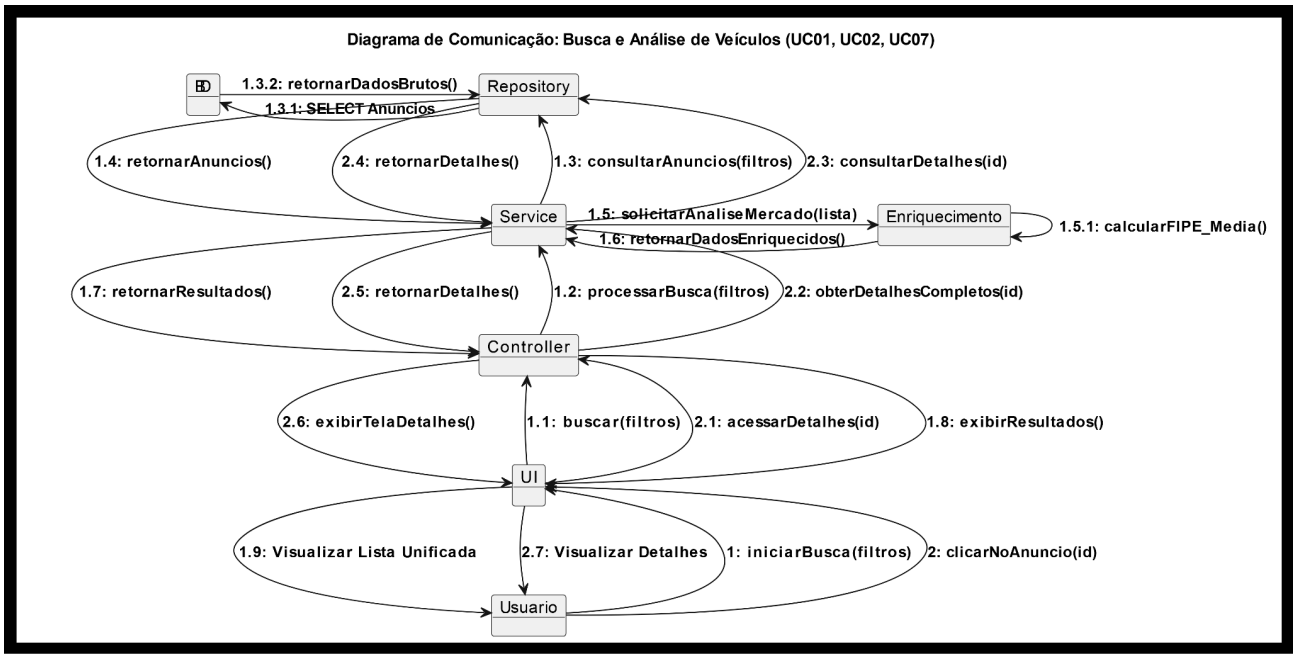

Figura 21. Diagrama de Comunicação de busca e análise de veículos

A Figura 21 representa o diagrama de comunicação responsável pelo fluxo de Gestão e Curadoria de Conteúdo. Nesse fluxo, o usuário Administrador inicia a comunicação por meio da Interface Admin para acessar as métricas (UC09), monitorar a coleta (UC10) ou realizar a validação e remoção de conteúdo (UC07/UC08). Essa comunicação é recebida pelo AdminController, que valida a permissão e define a ação. O AdminController se comunica com o AdminService, que processa a lógica de negócios: para moderação, ele interage com o AnuncioRepository para atualizar o *status* no Banco de Dados; para as métricas, ele se comunica com o Serviço de Monitoramento para obter logs e o *status* de saúde dos *scrapers*. Esse diagrama contempla os casos de uso UC07, UC08, UC09 e UC10.

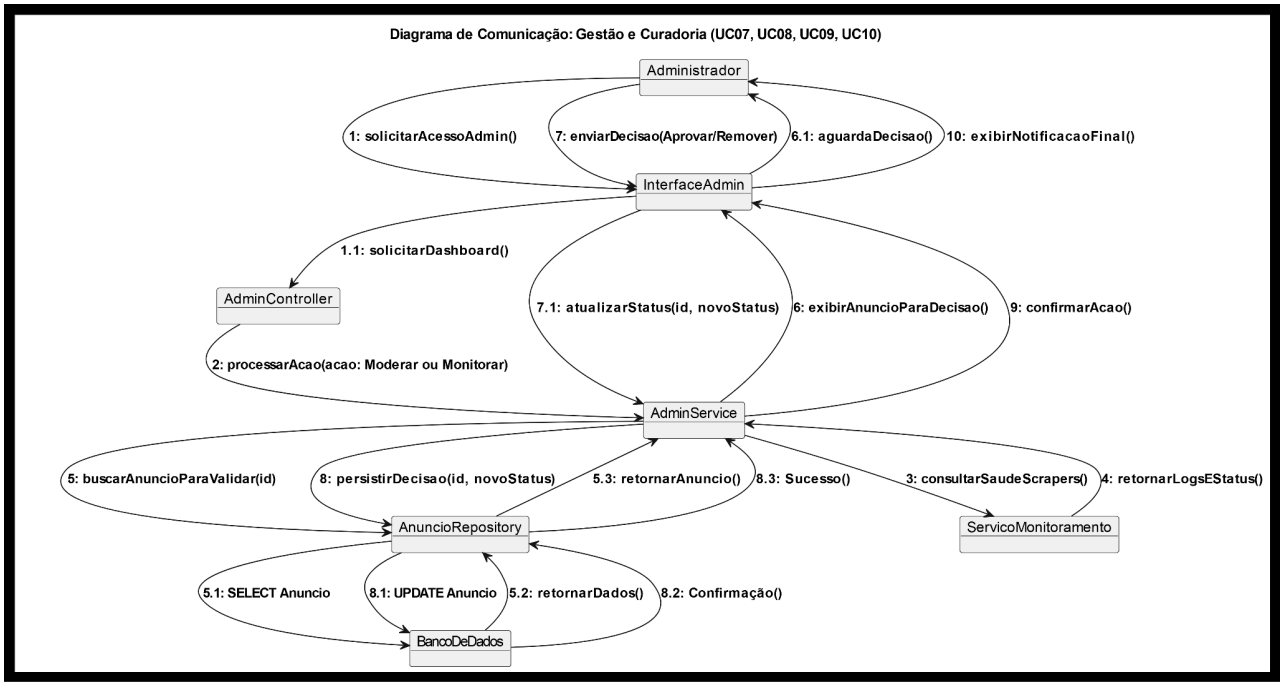

Figura 21. Diagrama de Comunicação de gestão e curadoria

A Figura 22 representa o diagrama de comunicação responsável pelo fluxo de Agregação e Persistência de Dados (*Scraper*). Nesse fluxo, o processo Job Agendado (ator assíncrono) inicia a comunicação com o *Controller* para disparar a rotina de coleta (UC11). O *Controller* orquestra o *Service*, que contém a lógica para interagir com a Fonte Externa (Marketplace) para obter os dados brutos. O *Service* normaliza o conteúdo e, em seguida, interage com o Banco de Dados para persistir os novos anúncios. Simultaneamente, o *Service* se comunica com o Serviço de Monitoramento para registrar o *status* da execução do Job. Esse diagrama contempla o caso de uso UC11.

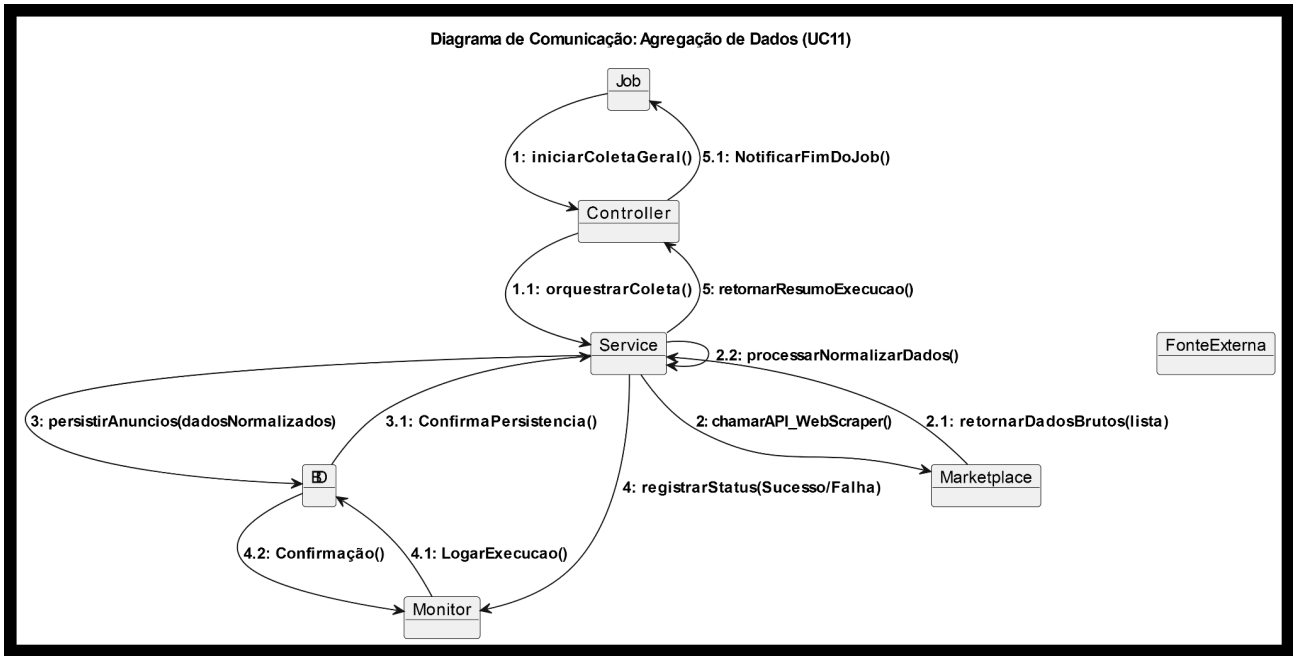

Figura 22. Diagrama de Comunicação de agregação de dados

3.4 Arquitetura

3.4.1 Motivação da stack e das ferramentas

A escolha da stack tecnológica do sistema BusCars fundamenta-se na necessidade de equilibrar desempenho, confiabilidade, segurança e facilidade de manutenção ao longo de todo o ciclo de vida do projeto. Considerando-se que se trata de uma plataforma web voltada à agregação de anúncios automotivos provenientes de diferentes fontes externas, tornou-se indispensável selecionar ferramentas capazes de lidar com grandes volumes de dados, prover mecanismos de cache e escalabilidade, além de garantir que o produto final seja acessível, estável e de fácil implantação em diferentes ambientes.

A linguagem *OCaml*, associada ao *framework Dream*, foi selecionada como base do desenvolvimento por possibilitar alto grau de segurança em tempo de compilação, expressividade e concisão no código-fonte. A tipagem forte e estática oferecida pelo *OCaml* contribui para a redução de erros em tempo de execução e promove maior robustez da aplicação, característica essencial em um sistema que realiza integração com múltiplos *marketplaces*. O *framework Dream*, por sua vez, viabiliza a construção de aplicações web de forma direta e moderna, com suporte a renderização de HTML e integração nativa a componentes de autenticação, roteamento e manipulação de requisições *HTTP*. A adoção dessa combinação também favorece a consistência técnica do projeto e a experimentação de paradigmas funcionais que simplificam a lógica de negócio.

O PostgreSQL foi escolhido como sistema gerenciador de banco de dados por se tratar de uma solução madura, estável e amplamente reconhecida na indústria. Entre seus principais diferenciais estão o suporte a dados semi estruturados (como JSONB), mecanismos avançados de

indexação (GIN e GiST) e extensões para geolocalização, recursos que se alinham às demandas de busca, filtragem e análise comparativa dos anúncios automotivos. Essas funcionalidades permitem ao sistema realizar consultas complexas de forma eficiente e fornecer indicadores enriquecidos, como valores médios de mercado e referências à Tabela FIPE.

Para lidar com operações de alta frequência, sessões de usuários e armazenamento temporário de resultados, foi adotado o Redis. A ferramenta desempenha um papel estratégico como mecanismo de cache e também como sistema de filas, permitindo processar integrações externas e normalização de dados de forma assíncrona, sem comprometer a responsividade da aplicação.

A utilização de *Docker* e *Docker Compose* justifica-se pela necessidade de padronizar os ambientes de desenvolvimento, homologação e produção. Essa abordagem assegura que todos os serviços (*frontend*, *backend*, banco de dados, *cache* e *proxy*) sejam executados de maneira consistente e portátil, reduzindo riscos de incompatibilidade entre diferentes sistemas operacionais ou configurações de servidores. Além disso, possibilita a rápida replicação do ambiente por qualquer membro da equipe, o que contribui para a agilidade do processo de desenvolvimento.

O Nginx foi incorporado à arquitetura como servidor *web* e *proxy* reverso, responsável pelo roteamento interno entre os contêineres, pela compressão de tráfego e pela otimização de conexões *HTTPS*. Em conjunto, a plataforma conta ainda com os recursos do *Cloudflare*, que atua como camada adicional de distribuição de conteúdo e segurança. A integração com a rede global da *Cloudflare* garante baixa latência, disponibilidade elevada e proteção contra ataques de negação de serviço (DDoS), oferecendo maior resiliência e confiabilidade ao sistema.

3.4.2 Descrição da arquitetura

De forma resumida, a arquitetura do BusCars pode ser representada conforme o diagrama abaixo:

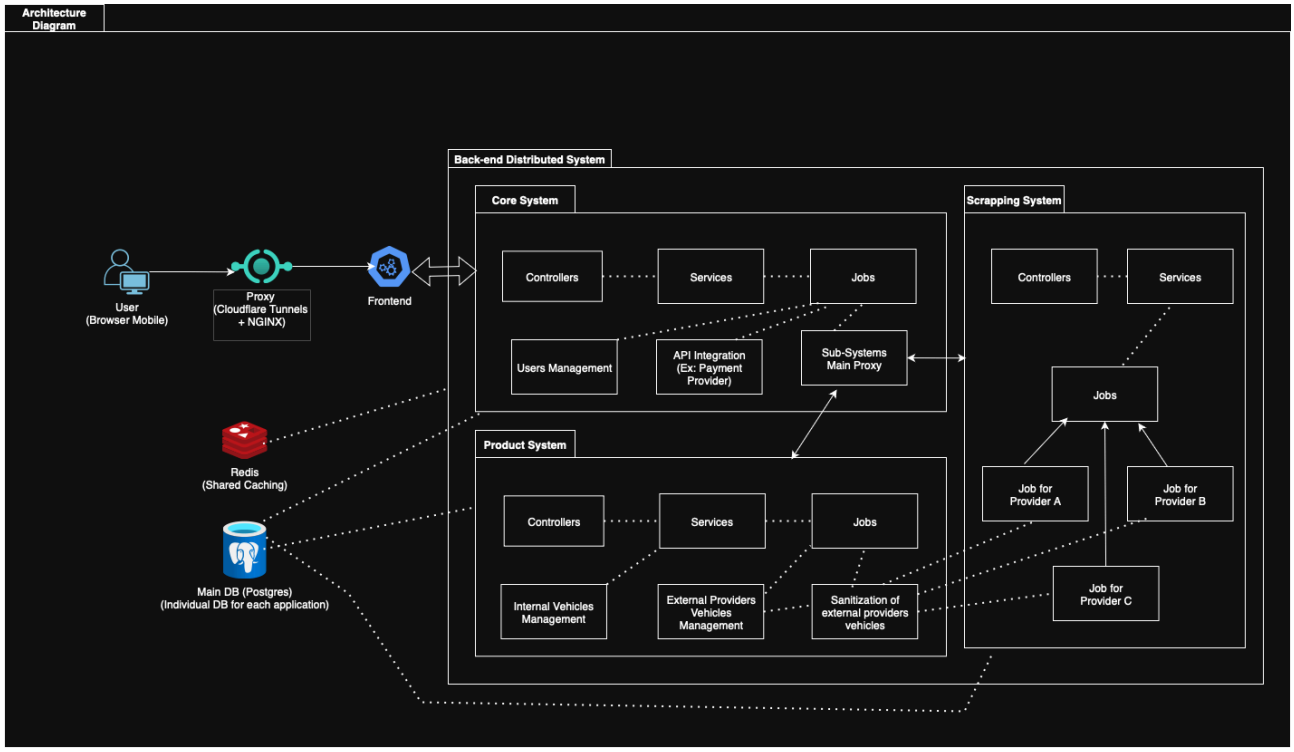

Figura 23. Diagrama de Arquitetura

A Figura 23 apresenta o diagrama de arquitetura, detalhando o sistema BusCars, a arquitetura foi concebida em camadas, de modo a assegurar a separação clara de responsabilidades, a manutenibilidade do código e a escalabilidade da solução. No nível superior, encontra-se a interface web, responsável por oferecer aos usuários finais uma experiência responsiva e intuitiva de busca e comparação de veículos. Essa interface é construída em HTML e CSS e entregue pelo *framework Dream*, que, além de gerenciar a camada de apresentação, também integra a lógica de controle das requisições.

No núcleo do sistema está o *backend*, igualmente desenvolvido em *OCaml* com o *framework Dream*. Ele desempenha o papel de camada de aplicação, recebendo requisições da interface, processando regras de negócio e interagindo com os serviços de persistência e *cache*. É nesta camada que se encontram as funcionalidades de agregação, normalização de dados, deduplicação de anúncios e cálculo de indicadores de mercado, assegurando que as informações apresentadas ao usuário sejam consistentes e enriquecidas.

O PostgreSQL constitui a camada de persistência, armazenando de forma estruturada todos os dados do sistema, incluindo usuários, anúncios importados de diferentes *marketplaces*, *logs* de integrações e métricas derivadas. O modelo de dados é projetado de forma a contemplar tanto atributos comuns (marca, modelo, ano, quilometragem, preço) quanto campos adicionais relacionados a padronização e enriquecimento das informações, permitindo consultas complexas e relatórios consolidados.

Complementarmente, o Redis exerce a função de *cache* e mecanismo de filas, contribuindo para a otimização da performance. Requisições de busca recorrentes podem ser rapidamente respondidas a partir do cache, enquanto filas são utilizadas para coordenar processos assíncronos, como a atualização de anúncios via *APIs* ou rotinas de *web scraping*. Essa abordagem evita sobrecarga no banco de dados principal e melhora a experiência do usuário, que recebe respostas mais rápidas.

Na camada de infraestrutura, o Nginx atua como proxy reverso, recebendo as conexões externas e direcionando-as adequadamente para os serviços internos executados em contêineres. Essa camada também implementa recursos de balanceamento de carga e compressão, além de interagir diretamente com os serviços de CDN e proteção fornecidos pela *Cloudflare*. O *Cloudflare*, por sua vez, garante resiliência, otimização de entrega de conteúdo e segurança contra ameaças externas, compondo a linha de frente da arquitetura.

Por fim, toda a solução é orquestrada por meio do *Docker Compose*, que descreve a configuração dos diferentes serviços, suas dependências e a rede interna de comunicação. Essa abordagem facilita a implantação tanto em servidores locais quanto em ambientes de nuvem, permitindo que a equipe mantenha controle sobre a escalabilidade e o monitoramento de cada componente.

3.5 Diagramas de Estados

O diagrama de estados representado na Figura 24, contém a lógica de mudança de estados que ocorrem durante o ciclo de vida de um Anúncio no BusCars. Nele, é possível observar que o anúncio passa por quatro estados principais: "Anúncio coletado", "Aguardando validação", "Visível para usuários" e "Excluído do sistema".

O ciclo começa no estado "Anúncio coletado", que representa o início do processamento após a ação do *scraper*. Em seguida, o anúncio entra no estado "Aguardando validação", onde ele é submetido ao crivo do Administrador. Este é um ponto crucial de decisão: se o anúncio for aprovado, ele segue para o estado "Visível para usuários", onde é exibido nas buscas e participa da análise de mercado. Se for rejeitado, ele é movido para o estado "Análise de conteúdo" para correção.

No estado "Visível para usuários", o anúncio pode ser analisado (transição para "análise de preço e filtros") e, posteriormente, passar para "Veículo vendido" e, finalmente, para "Fora de exibição" quando não for mais relevante. Os eventos principais envolvidos no processo incluem o *scraper* acionado, que insere o dado, a aprovação do Administrador, a solicitação de análise de mercado pelo usuário, e o evento Veículo vendido que o leva ao fim do ciclo.

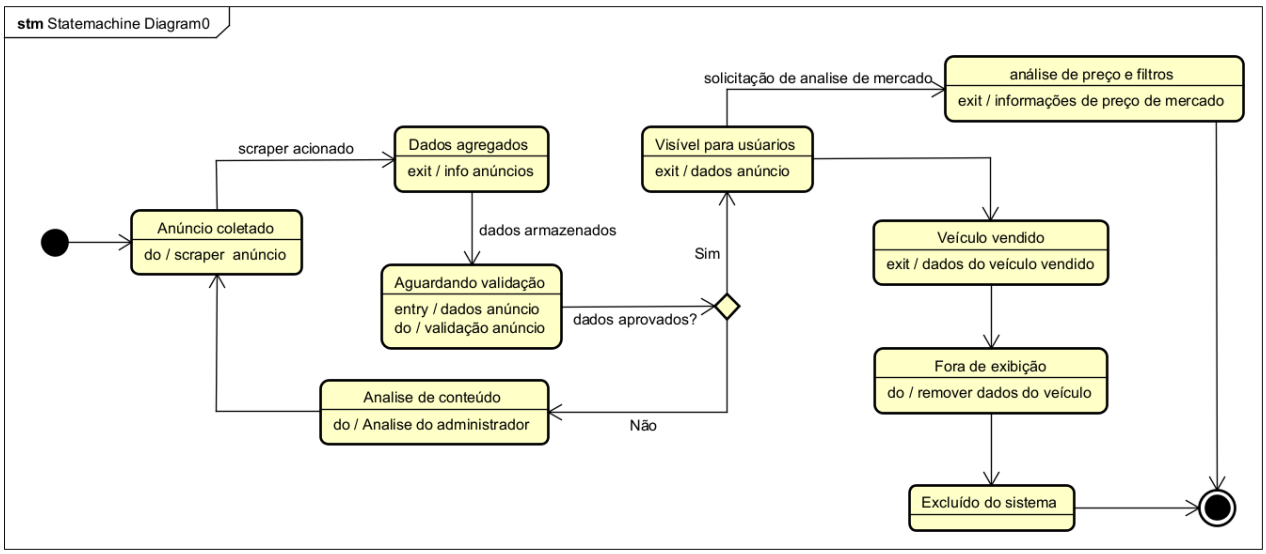

Figura 24. Diagrama de Estados

3.6 Diagrama de Componentes e Implantação.

Este item apresenta a visão estrutural (componentes e responsabilidades) e a visão física de execução (nós/contêineres e conexões) do BusCars. O Diagrama de Componentes (Seção 3.6.1) destaca módulos da aplicação (busca, agregação, enriquecimento, deduplicação, administração e métricas) e suas interações com *PostgreSQL*, *Redis*, *Nginx* e *Cloudflare*, além de integração com *marketplaces* e FIPE. O Diagrama de Implantação (Seção 3.6.2) descreve a topologia de execução em *Docker Compose*, com *App*/*Workers*/*DB*/*Cache* atrás de *Nginx* e *Cloudflare*.

3.6.1 Diagrama de Componentes

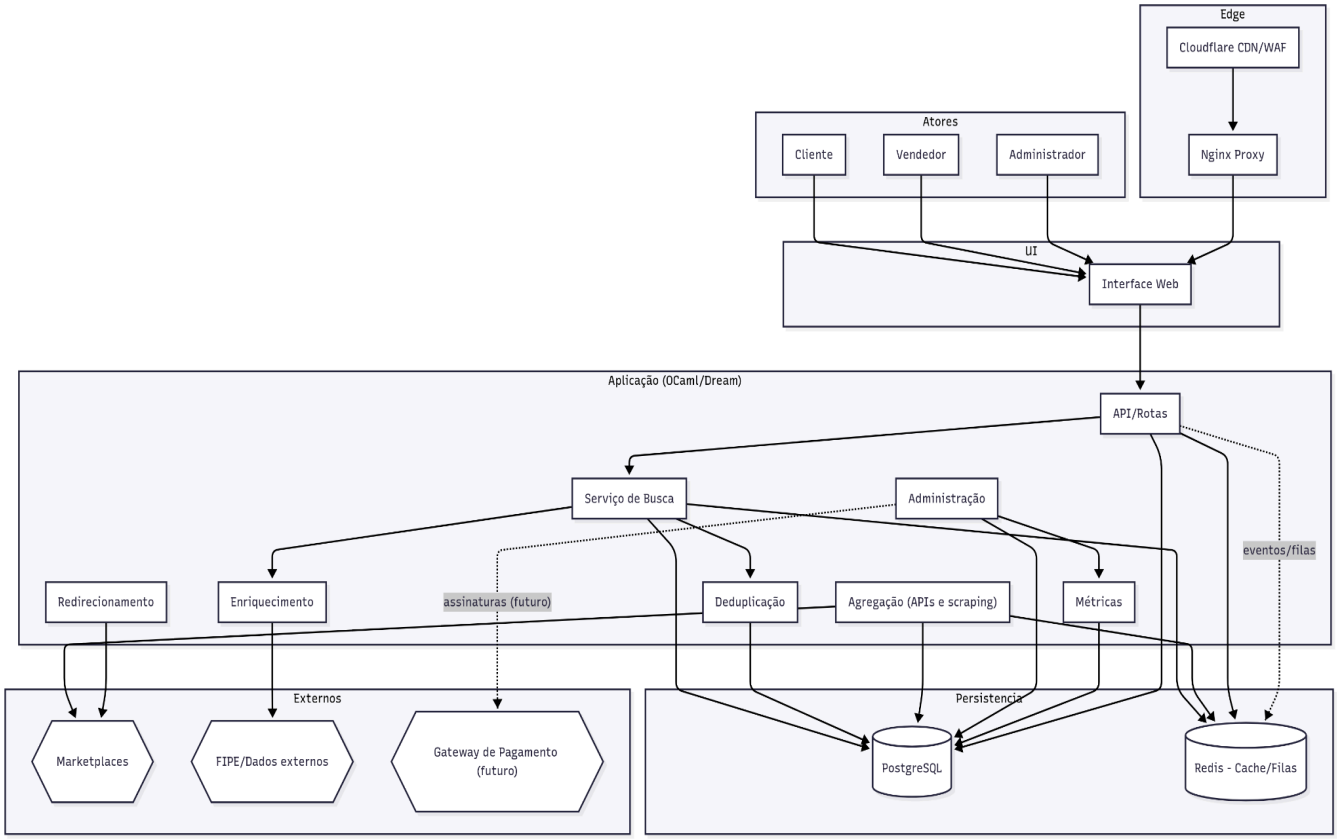

Figura 25. Diagrama de Componentes

A Figura 25 apresenta os principais componentes da solução e seus relacionamentos. Note que:

- a ***UI*** consome a ***API Core*** (*OCaml/Dream*);
- **Busca** consulta ***PostgreSQL*** e ***Redis*** e orquestra **Enriquecimento (FIPE)**, **Deduplicação** e **Agregação** (*APIs/scraping*);
- **Administração** e **Métricas** acessam a base para governança e observabilidade;
- Integrações de **pagamento** são tratadas como **evolução futura** (indicadas por arestas pontilhadas).

3.6.2 Diagrama de Implantação

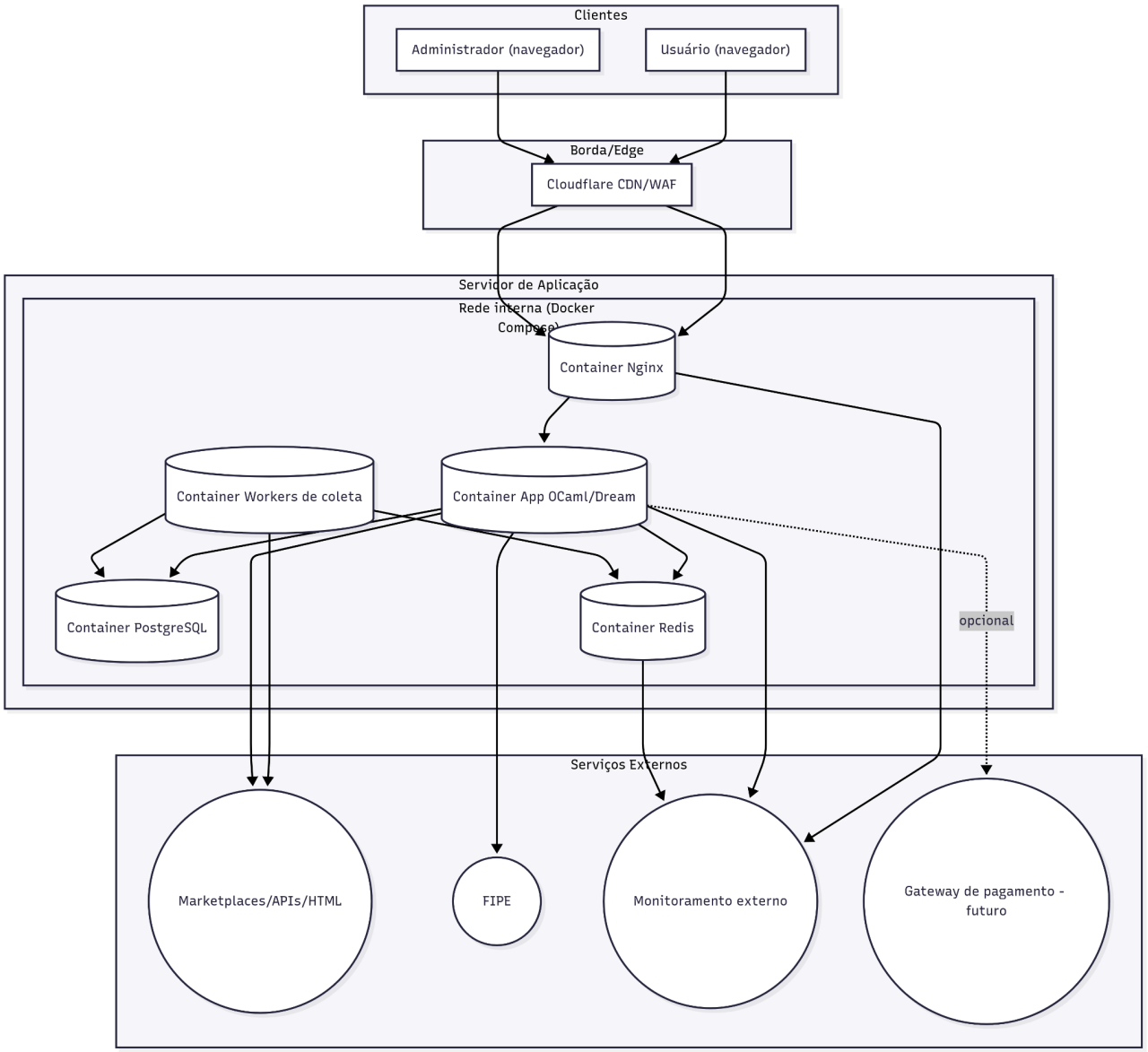

Figura 26. Diagrama de Implantação

A Figura 26 mostra a implantação em produção: *Cloudflare* → *Nginx* → *App*/*Workers*, com *PostgreSQL* e *Redis* na rede interna (*Compose*). *Workers* comunicam-se com os *marketplaces* (*APIs*/HTML) e o App consulta FIPE. Monitoramento externo observa *Nginx*/*App*/*Redis*. A integração com *gateway* de pagamento permanece planejada para fase posterior, por isso, aparece como “opcional”/pontilhada.

4. Projeto de Interface com Usuário

Esta seção tem como objetivo demonstrar e descrever as interfaces de interação com o usuário que compõem a aplicação. Para isso, foram utilizados *mockups* de um projeto demo. As interfaces foram correlacionadas aos casos de uso especificados na Seção 2.3.1, mapeando todas as funcionalidades necessárias para o atendimento dos requisitos estabelecidos.

4.1 Esboço das Interfaces Comuns a Todos os Atores

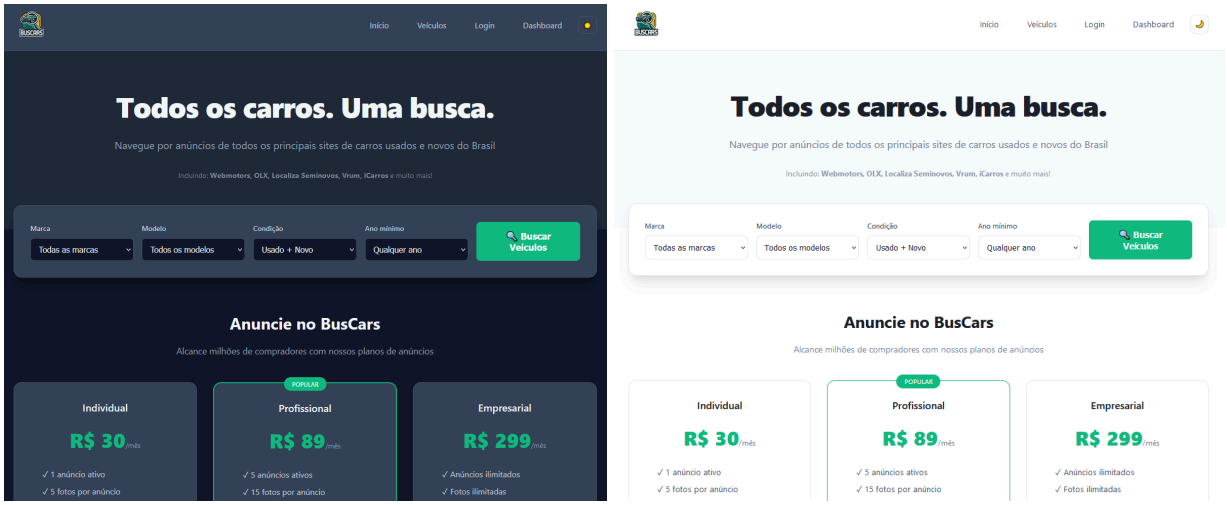

Figura 27. Tela inicial

A tela inicial, representada na Figura 27, do BusCar atua como o ponto de partida central para o usuário. No topo, um menu de navegação garante acesso rápido a outras seções da plataforma, como a área de login para usuários já cadastrados. O destaque da página, no entanto, é o buscador principal de veículos. Ele é posicionado de forma proeminente para convidar o usuário a iniciar sua pesquisa imediatamente, utilizando campos intuitivos para refinar a busca por marca, modelo ou ano.

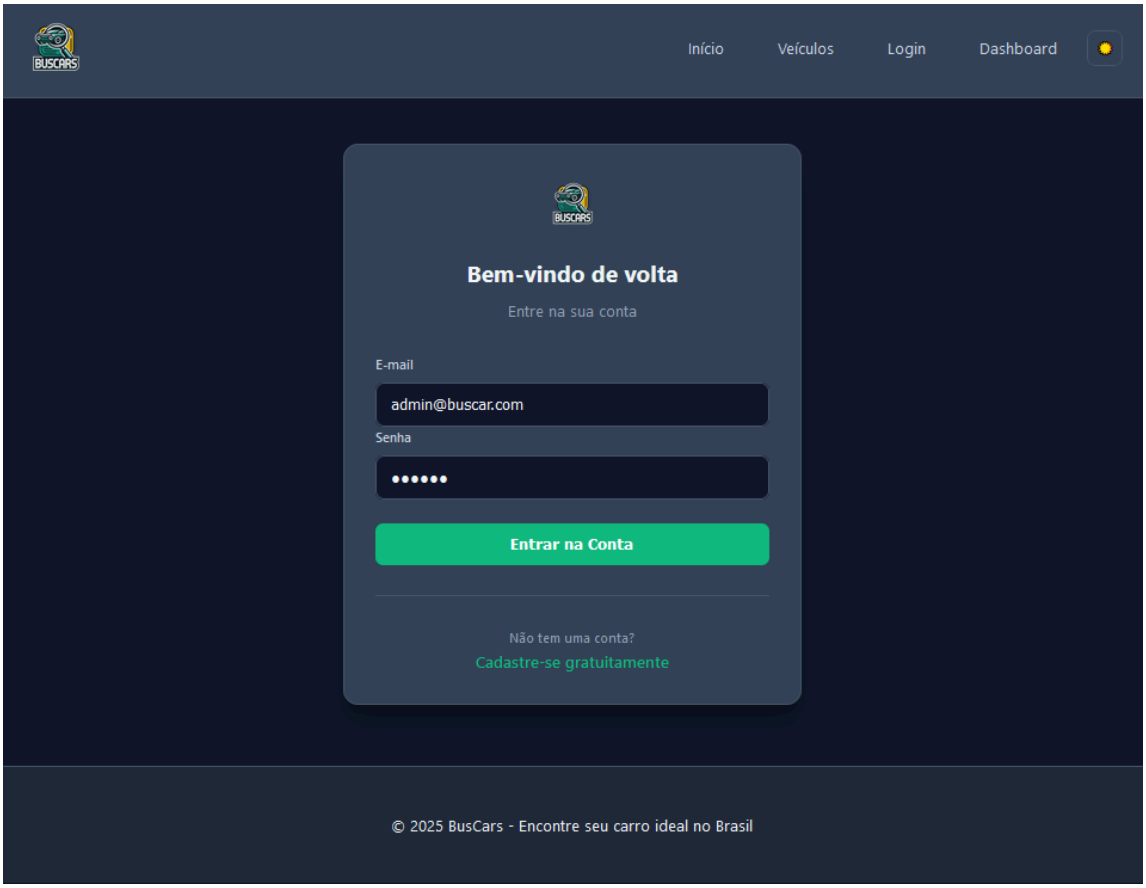

Figura 28. Tela de Login

A tela inicial, representada na Figura 28, serve como o ponto de entrada principal para a plataforma. No topo, ela oferece campos para login, permitindo que usuários já registrados acessem suas contas. No centro da tela, um campo de busca proeminente convida tanto o usuário logado quanto o visitante a iniciar uma pesquisa por veículos. O design é limpo e direto, priorizando a usabilidade para que a pesquisa comece imediatamente.

Figura 29. Tela de informações

A tela de informações e planos, representada na Figura 29, do BusCars foi criada para educar os usuários sobre o uso da plataforma e apresentar detalhes do funcionamento da ferramenta de busca e dos filtros para os compradores, enquanto para os vendedores, descreve os planos de assinatura disponíveis, com seus respectivos benefícios e custos, como a quantidade de anúncios e o acesso a ferramentas de análise. O objetivo principal é garantir total transparência sobre o modelo de negócio e as funcionalidades oferecidas em cada nível de serviço.

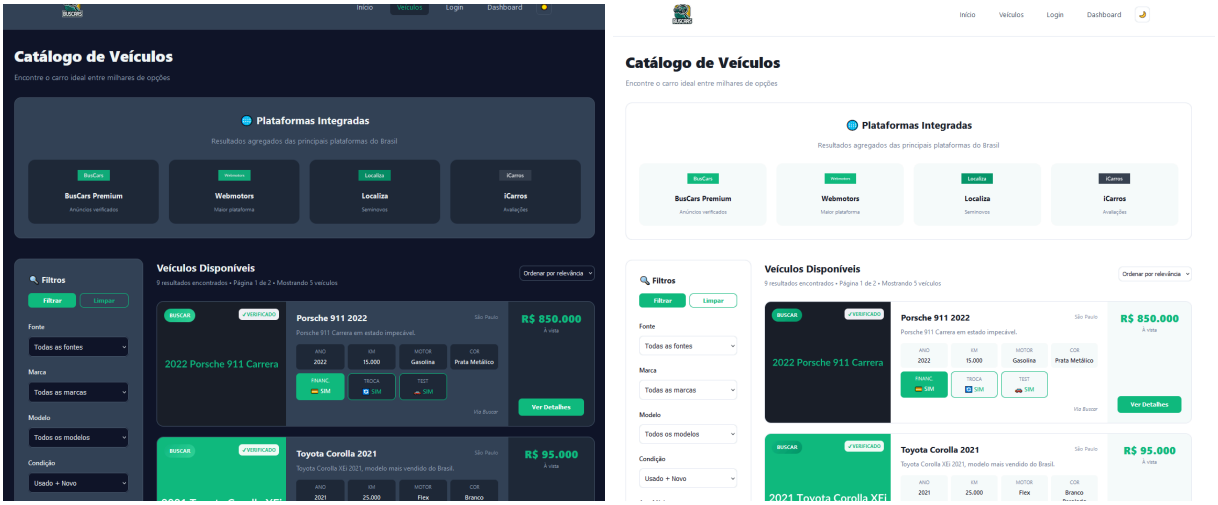

Figura 30. Tela de veículos

Esta tela de veículos, representada na Figura 30, é o coração da experiência de busca. Nela, os resultados da pesquisa são exibidos em um formato de grade ou lista. Cada cartão de anúncio apresenta as informações-chave de forma unificada e padronizada, independentemente da fonte original (como WebMotors ou Icarros). Essa visualização unificada é o principal diferencial do BusCars, facilitando a comparação visual entre diferentes veículos.

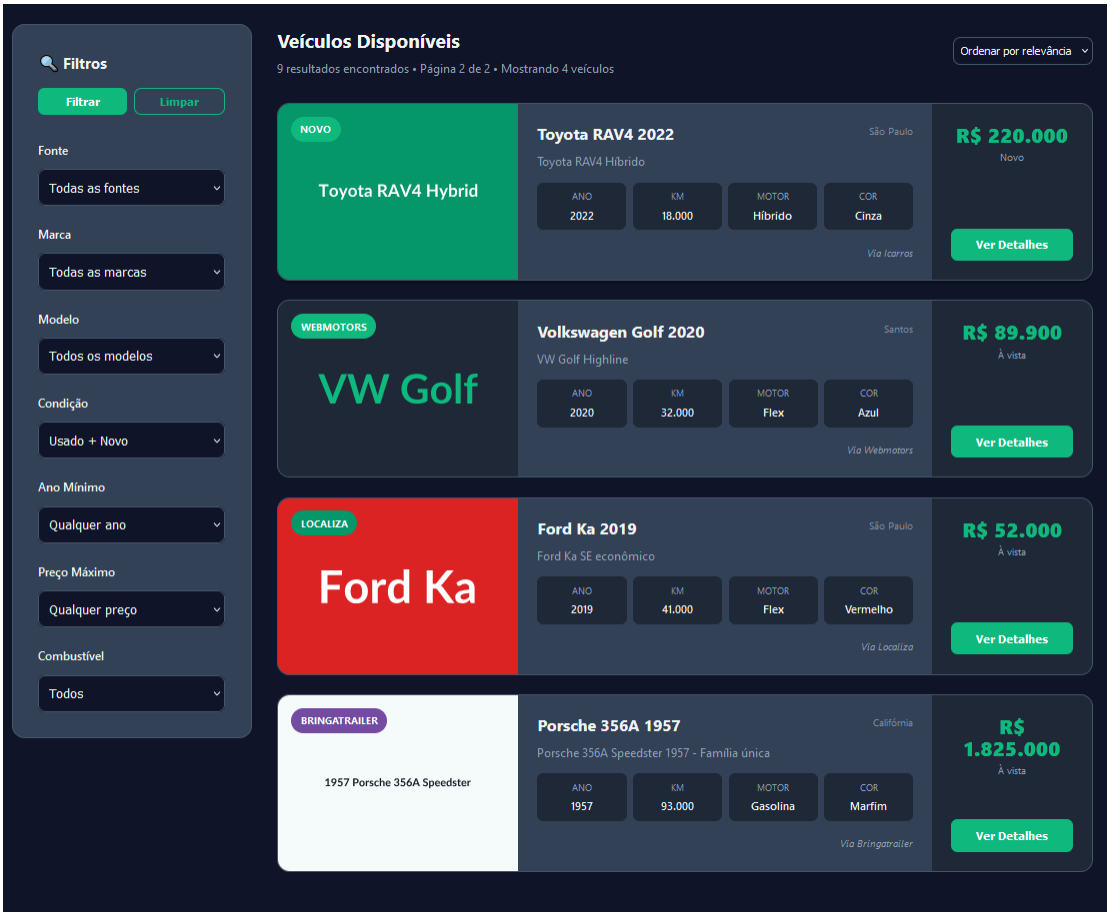

Figura 31. Tela de filtro de veículos

Após iniciar a busca, o usuário é direcionado para a tela de resultados, representada na Figura 31. Nela, a seção de filtros de busca se destaca como a ferramenta essencial para refinar a pesquisa. O usuário pode aplicar diversos critérios, como marca, modelo, ano, faixa de preço e quilometragem, para personalizar os resultados e encontrar exatamente o que procura, eliminando anúncios irrelevantes.

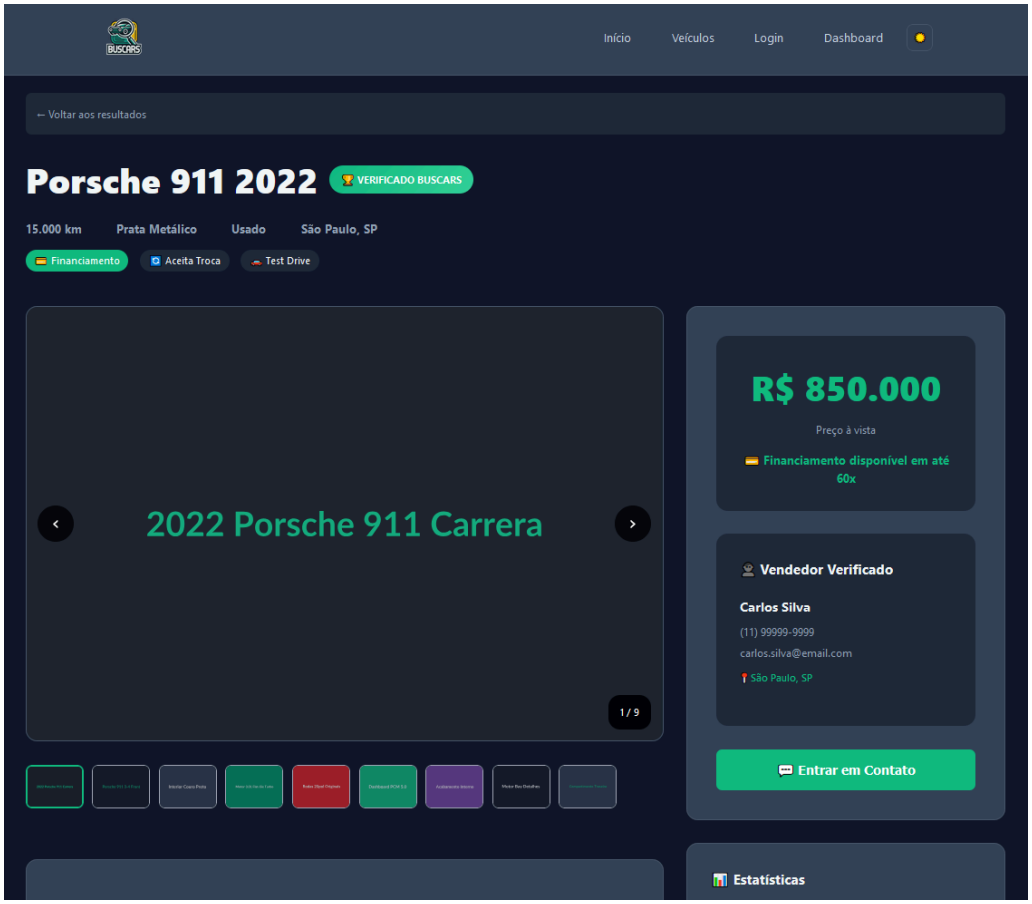

Figura 32. Tela de informações do veículo

Ao clicar em um veículo na tela de visualização, representada na Figura 32, o usuário acessa a página de detalhes completos. Aqui, todas as informações do anúncio são apresentadas de forma aprofundada. Além dos dados básicos como fotos, descrição e contato do vendedor, o sistema enriquece o conteúdo com informações adicionais, como o valor de referência da Tabela FIPE, ajudando o comprador a ter uma visão completa e segura sobre o valor do veículo.

4.2 Esboço das Interfaces Usadas pelo Ator Vendedor

Esta subseção apresenta as telas específicas para o Vendedor, ator que utiliza o sistema para acompanhar o mercado e, em fases futuras, publicar seus próprios anúncios com maior riqueza de detalhes. As telas desta seção são meramente demonstrativas.

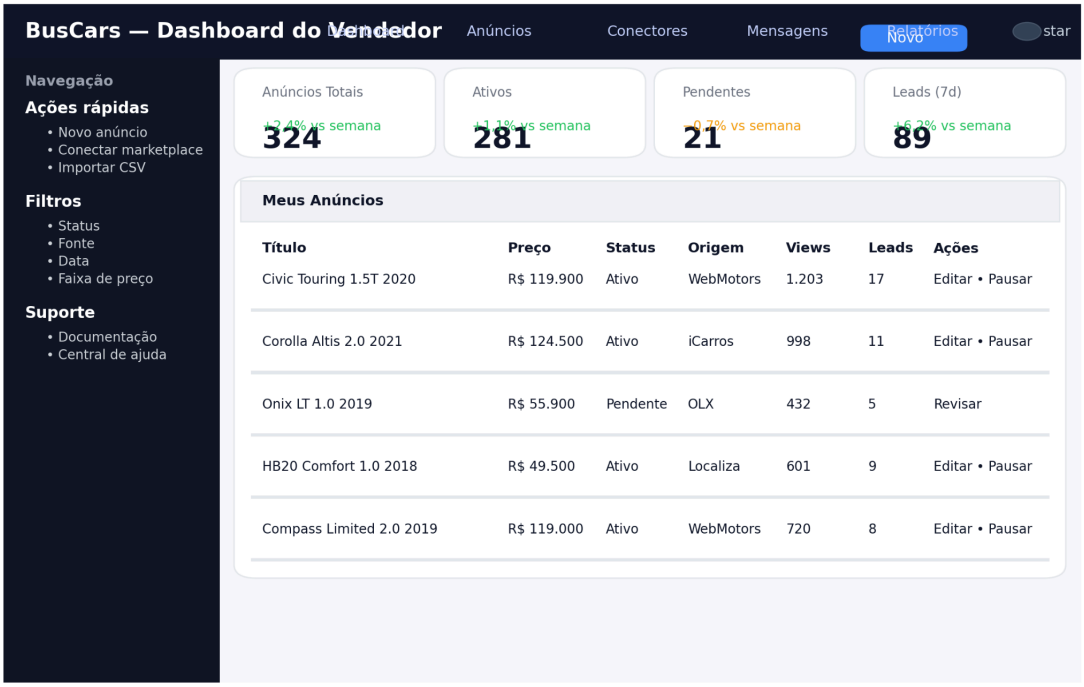

Figura 33. Tela de esboço do dashboard do Vendedor

4.3 Esboço das Interfaces Usadas pelo Ator Administrador

Esta subseção apresenta as telas do Administrador, responsável por operação/gestão do sistema: curadoria dos dados, monitoramento das coletas e eficiência da plataforma. As telas desta seção são meramente demonstrativas.

Figura 34. Tela de esboço do dashboard do Administrador

5. Glossário e Modelos de Dados

Nesta seção, é apresentado o glossário de atributos do sistema, representado pela Tabela 13. Nela, estão listados os atributos reconhecidos pelo usuário nas entradas de dados e nas saídas de dados do sistema, juntamente com suas descrições detalhadas. Cada atributo é acompanhado pelo seu formato correspondente e uma explicação que contextualiza seu uso e significado dentro do contexto do sistema BusCars. Este glossário é uma ferramenta essencial para entender e interpretar os dados manipulados pelo sistema, fornecendo uma referência clara e concisa para todos os atributos utilizados. Além disso, o glossário de atributos complementa a compreensão oferecida pelo Diagrama de Entidades Relacional na Figura 35, fornecendo uma visão detalhada dos elementos de dados e seus relacionamentos no contexto do sistema.

Atributo	**Formato**	**Descrição**
idAnuncio	INT	Identificador único para cada anúncio de veículo agregado ou publicado no sistema.
tituloAnuncio	VARCHAR	Título do anúncio, geralmente incluindo marca, modelo e ano do veículo.
descricaoAnuncio	VARCHAR	Descrição detalhada do veículo conforme coletada da fonte original.
precoAnuncio	FLOAT	Valor monetário do veículo anunciado.
dataColeta	DATETIME	Data e hora em que o anúncio foi coletado e inserido no sistema.
origemAnuncio	VARCHAR	Nome da plataforma de onde o anúncio foi coletado (ex: "WebMotors", "ICarros").
linkOriginal	VARCHAR	*URL* original do anúncio na plataforma de origem.
isDuplicado	BOOLEAN	Indicador booleano que sinaliza se o anúncio é uma duplicata já identificada.
idVeiculo	INT	Identificador único de cada veículo cadastrado no sistema.
marca	VARCHAR	Marca do veículo (ex: "Honda", "Volkswagen").
modelo	VARCHAR	Modelo do veículo (ex: "Civic", "Gol").
anoFabricacao	INT	Ano de fabricação do veículo.
quilometragem	FLOAT	Quilometragem atual do veículo.
combustivel	VARCHAR	Tipo de combustível do veículo (ex: "Gasolina", "Flex").
valorFIPE	FLOAT	Valor de referência do veículo na Tabela FIPE.
idUsuario	INT	Identificador único para cada usuário registrado no sistema (Cliente, Vendedor, Administrador).

nomeUsuario	VARCHAR	Nome completo ou de perfil do usuário.
email	VARCHAR	Endereço de e-mail do usuário, utilizado para login e comunicação.
senhaHash	VARCHAR	Senha do usuário criptografada para segurança.
tipoUsuario	VARCHAR	Categoria do usuário no sistema (ex: "Cliente", "Vendedor", "Administrador").
idFonteDados	INT	Identificador único para cada fonte de dados externa (plataforma de onde os anúncios são coletados).
nomeFonte	VARCHAR	Nome da plataforma de origem (ex: "WebMotors", "iCarros").
urlBaseFonte	VARCHAR	*URL* base da plataforma de origem.
statusColeta	VARCHAR	*Status* atual da rotina de coleta da fonte de dados (ex: "Sucesso", "Falha").
idBuscaSalva	INT	Identificador único para cada busca salva pelo usuário.
filtrosBusca	VARCHAR	*String* ou formato JSON contendo os critérios de filtro da busca salva.
dataBusca	DATETIME	Data em que a busca foi salva.
idFavorito	INT	Identificador único para um anúncio marcado como favorito por um usuário.

Tabela 13. Glossário de dados dos atributos do projeto

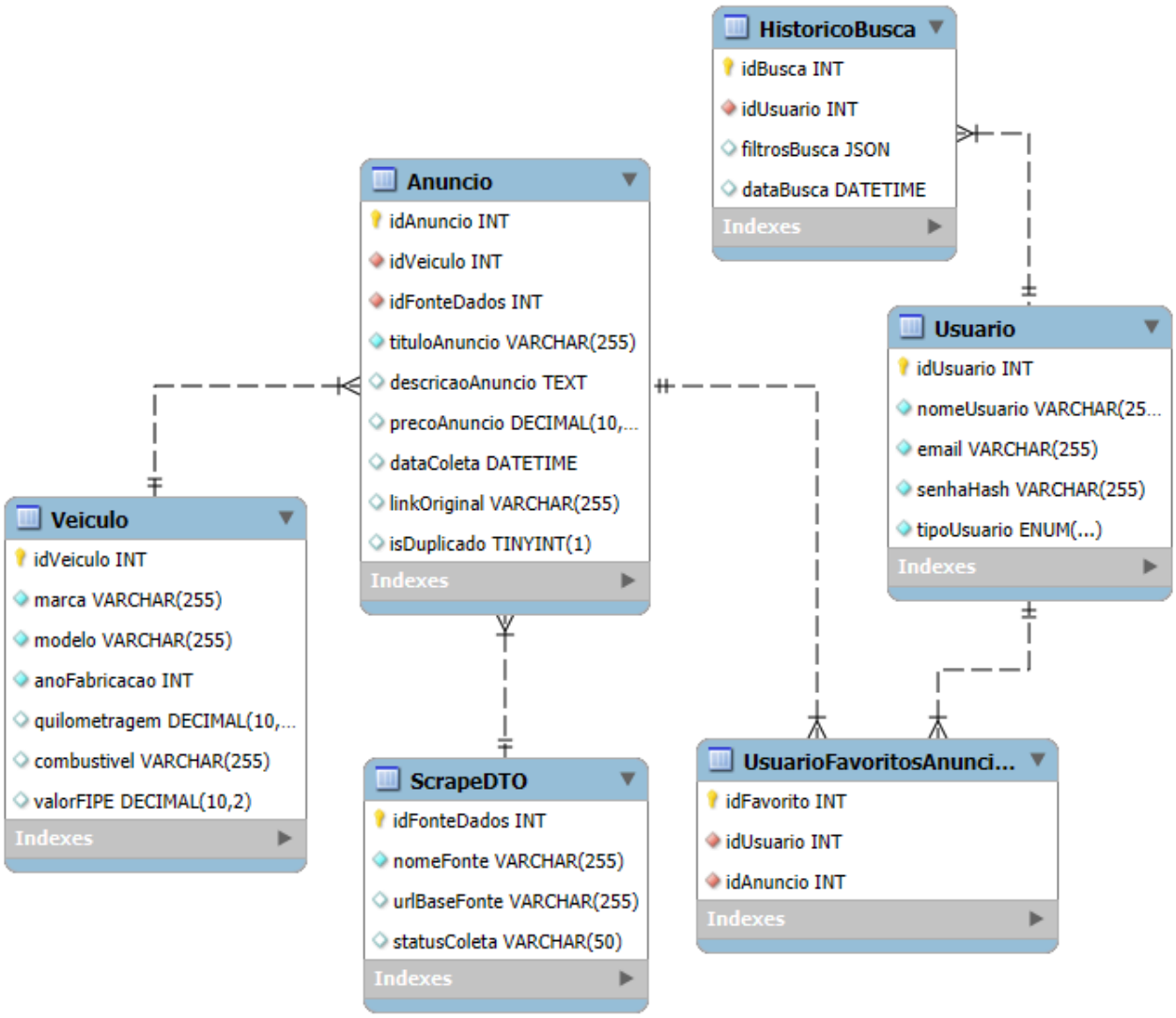

Figura 35. Diagrama de Entidades Relacional

6. Casos de Teste

Esta seção apresenta casos de teste de aceitação e integração realizados conforme a especificação do sistema BusCars. Na Seção 6.1, são detalhados os casos de teste de aceitação, elaborados a partir das histórias de usuário e fluxos principais (busca, detalhes, visualização unificada, enriquecimento/FIPE, deduplicação, curadoria, coleta e métricas). Na Seção 6.2, são descritos os casos de teste de integração, com foco na interação entre os módulos internos (*API*/Serviços, Normalização, *Dedup*, Cálculo de preços e Métricas) e serviços externos (*marketplaces* via *API*/HTML e FIPE). Cada caso explicita pré-condições, ações, entradas e o resultado esperado.

6.1 Testes de Aceitacao

Os testes de aceitação são destinados a garantir que o sistema BusCars atenda aos requisitos funcionais e não funcionais estabelecidos. Estes testes verificam se o sistema cumpre as expectativas dos usuários, e está pronto para ser lançado em produção. A seguir, são listadas essas necessidades, identificadas pela letra N e uma numeração única.

N1. Busca e Filtros
N2. Visualização Unificada
N3. Detalhes do Veículo
N4. Enriquecimento / Preço de Mercado (FIPE)
N5. Deduplicação de Anúncios
N6. Curadoria e Validação de Conteúdo
N7. Coleta e Monitoramento
N8. UX e Desempenho

N1. Busca e Filtros

Identificador	**N1TA1**
Necessidade	Busca e Filtros
Caso de Teste	Cliente busca veículos aplicando filtros
Pré-condições	• Base com anúncios normalizados e índices criados • Serviço de busca operacional
Dados de entrada	Filtros (ex: marca='Honda', modelo='Civic', ano >= 2018, uf='MG')
Ações	1) Acessar a página de busca; 2) Informar filtros; 3) Enviar busca; 4) Navegar paginação.
Dado de resposta	Lista de anúncios compatíveis com os filtros; paginação funcional; ordenação padrão aplicável.

Tabela 14. Teste de aceitação de Cliente busca veículos aplicando filtros.

Identificador	**N1TA2**
Necessidade	Busca e Filtros

Caso de Teste	Busca sem filtros retorna resultados padrão
Pré-condições	• Base com anúncios normalizados e índices criados • Serviço de busca operacional
Dados de entrada	Nenhum filtro aplicado.
Ações	1) Acessar a página de busca; 2) Submeter sem filtros.
Dado de resposta	Retorno de anúncios em ordem padrão definida (ex.: mais recentes ou relevância).

Tabela 15. Teste de aceitação de Busca sem filtros retorna resultados padrão.

N2. Visualização Unificada

Identificador	**N2TA1**
Necessidade	Visualização Unificada
Caso de Teste	Exibir todos os anúncios do mesmo veículo em página unificada
Pré-condições	• Existência de múltiplas fontes para o mesmo veículo. • Normalização concluída.
Dados de entrada	ID do veículo consolidado.
Ações	1) A partir da lista, abrir 'Ver todos'; 2) Conferir ofertas consolidadas.
Dado de resposta	Página exibe ofertas de múltiplas fontes para o mesmo veículo sem duplicatas redundantes.

Tabela 16. Teste de aceitação de Exibir todos os anúncios do mesmo veículo em página unificada.

N3. Detalhes do Veículo

Identificador	**N3TA1**
Necessidade	Detalhes do Veículo
Caso de Teste	Acessar detalhes do veículo a partir dos resultados

Pré-condições	Registro possui campos essenciais e enriquecimento disponível.
Dados de entrada	ID de anúncio.
Ações	1) Realizar busca; 2) Clicar em um resultado; 3) Validar campos exibidos.
Dado de resposta	Ficha técnica exibida com dados consolidados e normalizados.

Tabela 17. Teste de aceitação de Acessar detalhes do veículo a partir dos resultados.

N4. Exibir indicador de preço vs. mercado (FIPE/faixa)

Identificador	**N4TA1**
Necessidade	Enriquecimento / Preço de Mercado (FIPE)
Caso de Teste	Exibir indicador de preço vs. mercado (FIPE/faixa)
Pré-condições	• Serviço de enriquecimento/estatística operacional • Dados FIPE disponíveis.
Dados de entrada	Modelo/Ano do veículo.
Ações	1) Abrir detalhes ou visão unificada; 2) Ver indicador de preço relativo.
Dado de resposta	Indicador 'abaixo/na média/acima' coerente com FIPE/faixa calculada.

Tabela 18. Teste de aceitação de Exibir indicador de preço vs. mercado (FIPE/faixa).

N5. Deduplicação de Anúncios

Identificador	**N5TA1**
Necessidade	Deduplicação de Anúncios
Caso de Teste	Identificar e ocultar duplicados antes da exibição
Pré-condições	Registros duplicados conhecidos estão

	presentes na base.
Dados de entrada	Lista de resultados com potenciais duplicados.
Ações	1) Executar busca; 2) Conferir que duplicatas são unificadas ou ocultas.
Dado de resposta	Não existem repetições desnecessárias do mesmo veículo, e/ou unificação aplicada.

Tabela 19. Teste de aceitação de Lista de resultados com potenciais duplicados.

N6. Curadoria e Validação de Conteúdo

Identificador	**N6TA1**
Necessidade	Curadoria e Validação de Conteúdo
Caso de Teste	Bloquear anúncio com dados inconsistentes/fraudulentos
Pré-condições	Registro inconsistente presente; regras de curadoria definidas.
Dados de entrada	Anúncio com dados inválidos (ex.: preço 0, campos essenciais ausentes).
Ações	1) Executar curadoria; 2) Verificar lista/detalhe.
Dado de resposta	Anúncio inválido não é exibido em busca/detalhes; *log*/justificativa registrados.

Tabela 20. Teste de aceitação de Bloquear anúncio com dados inconsistentes/fraudulentos.

N7. Coleta e Monitoramento

Identificador	**N7TA1**
Necessidade	Coleta e Monitoramento
Caso de Teste	Painel exibe *status* de coleta por fonte
Pré-condições	Coletas executadas previamente; *status* persistidos.

Dados de entrada	Acesso ao painel administrativo.
Ações	1) Abrir painel; 2) Visualizar *status* por fonte; 3) Filtrar por data.
Dado de resposta	*Status* das coletas (sucesso/falha, data/hora, contagens) visíveis e consistentes.

Tabela 21. Teste de aceitação de Painel exibe status de coleta por fonte.

Identificador	**N7TA2**
Necessidade	Coleta e Monitoramento
Caso de Teste	Registrar e visualizar falhas de coleta
Pré-condições	Falhas simuladas ou ocorridas; telemetria habilitada.
Dados de entrada	Acesso ao painel administrativo.
Ações	1) Abrir painel; 2) Ver seção de falhas; 3) Validar mensagens/*logs*.
Dado de resposta	Falhas aparecem com fonte, horário e causa; sem impacto na disponibilidade do restante.

Tabela 22. Teste de aceitação de Registrar e visualizar falhas de coleta.

N8. UX e Desempenho

Identificador	**N8TA1**
Necessidade	UX e Desempenho
Caso de Teste	Tempo de resposta da busca dentro do limite
Pré-condições	• Ambiente estável; • Telemetria habilitada; • Base com dados variados e em grandes quantidades (ex: >=1000)
Dados de entrada	Execução de busca típica (ex.: 4 filtros)
Ações	1) Executar busca;

	2) Medir latência ponta-a-ponta.
Dado de resposta	Tempo de resposta <= 2s (meta de UX) em 95% das execuções.

Tabela 23. Teste de aceitação de Tempo de resposta da busca dentro do limite.

Identificador	**N8TA2**
Necessidade	UX e Desempenho
Caso de Teste	Consistência na aplicação de filtros e paginação
Pré-condições	• Ambiente estável; • Telemetria habilitada; • Base com dados variados e em grandes quantidades (ex: >=1000)
Dados de entrada	Sequência de filtros + mudança de página; alteração/remoção de filtros.
Ações	1) Aplicar filtros; 2) Paginar; 3) Alterar filtros; 4) Ver resultados.
Dado de resposta	Resultados sempre coerentes com filtros ativos; paginação preserva contexto.

Tabela 24. Teste de aceitação de Consistência na aplicação de filtros e paginação.

6.2 Testes de Integração

Os testes de integração validam o funcionamento conjunto entre camadas internas e serviços externos, e por consequência, são cruciais para assegurar que os diferentes módulos e componentes do sistema Bus Cars operem em harmonia, proporcionando uma experiência de usuário fluida e confiável. A seguir, são apresentados dez casos de testes de integração detalhados.

N1. Busca e Filtros

Identificador	**N1TI1**
Necessidade	Busca e Filtros

Caso de Teste	Integração *API* de Busca ↔ *PostgreSQL* (filtros e ordenação)
Sistemas Envolvidos	• BusCars *API* (Busca) • PostgreSQL
Interface	Requisições *HTTP* → Camada de Persistência (SQL)
Pré-condições	• Esquema e índices criados; • base populada; • *API* acessível.
Entradas	Filtros variados (marca, modelo, ano, preço, UF); paginação; ordenação.
Resultados Esperados	Resultados corretos e estáveis, com performance adequada e paginação/ordenação consistentes.

Tabela 25. Teste de Integração de Integração API de Busca ↔ PostgreSQL (filtros e ordenação)

Identificador	**N1TI2**
Necessidade	Busca e Filtros
Caso de Teste	Integração Busca ↔ *Redis* (*cache*/filas)
Sistemas Envolvidos	• BusCars *API* (Busca) • *Redis* (*cache*/filas)
Interface	Acesso a cache/filas (*driver Redis*)
Pré-condições	• *Redis* disponível; • Chaves de *cache* habilitadas; • Políticas de expiração configuradas.
Entradas	Execuções repetidas da mesma consulta; eventos de invalidação
Resultados Esperados	Acertos de cache para consultas idempotentes; invalidação correta ao mudar filtros/dados.

Tabela 26. Teste de Integração de Integração Busca ↔ Redis (cache/filas)

N2. Visualização Unificada

Identificador	N2TI1
Necessidade	Visualização Unificada
Caso de Teste	Normalização/Consolidação ↔ Página Unificada
Sistemas Envolvidos	• Serviço de Normalização/Consolidação • BusCars *API*/*Frontend*
Interface	Jobs/processos assíncronos + *API* interna
Pré-condições	• Dados de múltiplas fontes para o mesmo veículo. • Regras de normalização definidas.
Entradas	• Execução da rotina de normalização • Requisição de exibição unificada.
Resultados Esperados	Dados padronizados e consolidados; unificação refletida na página (sem duplicatas).

Tabela 27. Teste de Integração de Integração Visualização Unificada

N3. Detalhes do Veículo

Identificador	N3TI1
Necessidade	Detalhes do Veículo
Caso de Teste	Detalhes ↔ Enriquecimento (FIPE)
Sistemas Envolvidos	• Serviço de Detalhes, Serviço de Enriquecimento, FIPE
Interface	Chamada ***HTTP*** para FIPE (ou *dataset* interno), ***API*** interna entre serviços
Pré-condições	• Chaves/credenciais configuradas • Conectividade à FIPE • Dados mínimos do veículo disponíveis.
Entradas	• Consulta por ID de anúncio/modelo/ano.
Resultados Esperados	Ficha traz campos enriquecidos (ex.: faixa/valor FIPE) de forma consistente e

	resiliente.

Tabela 28. Teste de Integração de Integração Detalhes do Veículo

N4. Exibir indicador de preço vs. mercado (FIPE/faixa)

Identificador	**N4TI1**
Necessidade	Enriquecimento / Preço de Mercado (FIPE)
Caso de Teste	Cálculo de preço médio ↔ FIPE ↔ BD
Sistemas Envolvidos	• Serviço de Estatística/Agregação, FIPE, ***PostgreSQL***
Interface	Jobs + *HTTP* FIPE + SQL
Pré-condições	• Agendador habilitado; credenciais FIPE válidas • Tabelas agregadas existentes.
Entradas	• Execução do cálculo (*batch*) por modelo/ano • Atualização incremental.
Resultados Esperados	Tabelas agregadas atualizadas; indicador de preço funcionando para os cenários está relacionados.

Tabela 29. Teste de Integração de Integração Exibir indicador de preço vs. mercado

N5. Deduplicação de Anúncios

Identificador	**N5TI1**
Necessidade	Deduplicação de Anúncios
Caso de Teste	Pipeline de Deduplicação ↔ Unificação
Sistemas Envolvidos	• Serviço de Deduplicação, Normalização/Consolidação, *PostgreSQL*
Interface	Jobs assíncronos + *SQL*
Pré-condições	• Regras/heurísticas de *dedup* definidas • Base com duplicatas conhecidas.

Entradas	• Execução da rotina de *dedup* • Leitura posterior via página unificada.
Resultados Esperados	Duplicatas marcadas e agrupadas; somente entradas únicas aparecem na interface.

Tabela 30. Teste de Integração de Integração Deduplicação de Anúncios

N6. Curadoria e Validação de Conteúdo

Identificador	**N6TI1**
Necessidade	Curadoria e Validação de Conteúdo
Caso de Teste	Curadoria/Validação ↔ *API*/Busca
Sistemas Envolvidos	• Serviço de Curadoria, BusCars *API*
Interface	*API* interna e *flags* de visibilidade
Pré-condições	• Regras de curadoria configuradas • Registros inválidos na base de teste.
Entradas	• Execução de curadoria • Consultas subsequentes de busca/detalhes.
Resultados Esperados	Registros inválidos deixam de ser exibidos; justificativas/registros são persistidos.

Tabela 31. Teste de Integração de Integração Curadoria e Validação de Conteúdo

N7. Coleta e Monitoramento

Identificador	**N7TI1**
Necessidade	Coleta e Monitoramento
Caso de Teste	Coletor (*Scraping*/*API*) ↔ *BD* (*status*) ↔ Painel Admin
Sistemas Envolvidos	• Coletor de Fontes, *PostgreSQL*, Painel Admin
Interface	Jobs/HTTP + *SQL* + *UI*
Pré-condições	• *Jobs* configurados; fontes acessíveis

	• Tabelas de *status* e logs presentes.
Entradas	• Execução do coletor para múltiplas fontes • Abertura do painel.
Resultados Esperados	*Status* por fonte/execução registrados e exibidos no painel com horário e contagens.

Tabela 32. Teste de Integração de Integração Coleta e Monitoramento Jobs/HTTP

Identificador	**N7TI2**
Necessidade	Coleta e Monitoramento
Caso de Teste	Tratamento de falhas de coleta ↔ Telemetria
Sistemas Envolvidos	• Coletor de Fontes, Logger/Métricas, Painel Admin
Interface	Logs/telemetria + *UI*
Pré-condições	• Telemetria habilitada • Simulação de *timeouts*/erros *HTTP*.
Entradas	• Forçar falhas (*timeout*, 429/500, *HTML* inesperado) • Consultar painel/*logs*.
Resultados Esperados	Falhas registradas com causa; sistema resiliente; alerta/visibilidade para o Admin.

Tabela 33. Teste de Integração de Integração Coleta e Monitoramento Logs/Telemetria

N8. UX e Desempenho

Identificador	**N8TI1**
Necessidade	UX e Desempenho
Caso de Teste	Telemetria de busca ponta-a-ponta
Sistemas Envolvidos	• BusCars *API*, Enriquecimento, *PostgreSQL*, Métricas
Interface	Instrumentação/telemetria

Pré-condições	• Coletores de métricas ativos • *Thresholds* definidos.
Entradas	• Execução de buscas típicas e pesadas • Coleta de latência em cada estágio.
Resultados Esperados	Métricas de latência e taxa de erro consolidadas; relatórios para otimização contínua.

Tabela 34. Teste de Integração de Integração UX e Desempenho

7. Cronograma e Processo de Implementação

O desenvolvimento do projeto está programado para ocorrer ao longo de quatro meses e meio, com início em agosto de 2025 e conclusão prevista para a segunda quinzena de dezembro de 2025. As atividades de desenvolvimento foram organizadas em *issues* e divididas em *milestones*, gerenciadas por meio do *GitHub* do projeto. O processo de desenvolvimento será estruturado em *sprints*, cada um com aproximadamente 15 dias de duração. Ao final de cada iteração, haverá uma reunião de alinhamento com o stakeholder para fornecer feedback e realizar eventuais ajustes no escopo do projeto.

Milestones	**Luis Gustavo**	**Lucas Lima**
1ª Quinzena de Agosto	• Criação e organização dos boards no *GitHub* para o gerenciamento das tarefas. • Provisionamento e configuração da *VPS* (*Virtual Private Server*) para o ambiente de testes. • Criação do repositório *Git* e configuração de *branches*.	• Configuração do ambiente de desenvolvimento (*IDE*, *Docker*). • Definição da arquitetura inicial do sistema.
2ª Quinzena de Agosto	• Revisão e definição do modelo de dados para a agregação de veículos. • Criação das tabelas no banco de dados (*PostgreSQL*) para armazenar os dados dos anúncios.	• Implementação da estrutura inicial do projeto (*backend* e *frontend*). • Documentação inicial do ambiente configurado.
1ª Quinzena de Setembro	• Desenvolvimento de rotinas de *web scraping* e/ou integração	• Criação de endpoints de *API* para receber e processar os dados dos

	com *APIs* para a coleta de anúncios de diferentes *marketplaces*.	agregadores. • Implementação das regras de negócio para a normalização e padronização dos dados coletados.
1ª Quinzena de Setembro	• Desenvolvimento da interface de busca principal do sistema. • Implementação dos filtros avançados (marca, modelo, ano, preço, etc.).	• Desenvolvimento da visualização unificada dos anúncios. • Criação da página de detalhes dos veículos agregados.
1ª Quinzena de Outubro	• Implementação do enriquecimento de dados (exibição de Tabela FIPE). • Desenvolvimento do algoritmo para identificação de anúncios duplicados.	• Implementação da funcionalidade de análise de preços de mercado para o Vendedor. • Configuração da autenticação de usuários e desenvolvimento das interfaces de login e registro (para permitir futuras funcionalidades).
2ª Quinzena de Outubro	• Implementação do painel de administração para monitorar a coleta de dados e a saúde do sistema. • Desenvolvimento da interface para validação e moderação de conteúdo (remoção de duplicatas).	• Implementação de ferramentas de monitoramento (*Grafana*, *Elastic Stack*) para o *backend*. • Implementação de testes unitários para as funcionalidades principais.
1ª Quinzena de Novembro	• Implementação de melhorias de desempenho (*caching*, otimizações de banco de dados). • Configuração de dashboards e alertas para o monitoramento contínuo.	• Realização de testes de aceitação para garantir que o MVP atende aos requisitos do projeto. • Correções de bugs identificados nos testes e na etapa de desenvolvimento.
2ª Quinzena de Novembro	• Desenvolvimento da estrutura inicial para a futura funcionalidade de anúncios próprios (modelos de dados e endpoints). • Criação de dashboards e relatórios para a visualização das métricas do sistema.	• Implementação de feedback e avaliações de serviços (para futuras interações entre usuários). • Realização de testes de carga e desempenho.
1ª Quinzena	• *Deploy* da aplicação em	• Revisão e refinamento final do

de Dezembro	ambiente de produção. • Containerização da aplicação (*Docker*) e preparação para o *deploy* final.	código. • Verificação final do sistema em produção. • Criação e atualização da documentação técnica e de usuário (*Swagger*, *Markdown*).

Tabela 35. Cronograma de atividades do projeto