
Documentação de Projeto

para o sistema

MobU

Versão 1.0

Projeto de sistema elaborado pelo(s) aluno(s) Joaquim de Moura Thomaz Neto e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo dos professores Danilo de Quadros Maia Filho, Leonardo Vilela Cardoso, Raphael Ramos Dias Costa, orientação acadêmica do professor Cleiton Silva Tavares e orientação de TCC II do professor (a ser definido no próximo semestre).

14/09/2025

Tabela de Conteúdo

Tabela de Conteúdo	ii
Histórico de Revisões	ii
1. Introdução	1
2. Modelo de Requisitos	1
1.1 Descrição de Atores	1
1.2 Modelo de Casos de Uso	4
3. Modelo de Projeto	11
2.1 Diagrama de Classes	11
2.2 Diagramas de Sequência	15
2.3 Diagramas de Comunicação	17
2.4 Arquitetura Lógica: Diagramas de Pacotes	21
2.5 Diagramas de Estados	1
2.6 Diagrama de Componentes	1
4. Projeto de Interface com Usuário	2
3.1 Interfaces Comuns a Todos os Atores	2
3.2 Interfaces Usadas pelo Ator <A>	2
3.3 Interfaces Usadas pelo Ator 	2
5. Modelo de Dados	2
6. Modelo de Teste	2

Histórico de Revisões

1. Introdução

Este documento agrega: 1) a elaboração e revisão dos modelos de domínio e 2) os modelos de projeto para o sistema MobU. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema, que acompanha este documento. Anexo a este documento também se encontra o Glossário.

O sistema MobU tem como proposta o desenvolvimento de uma plataforma de mobilidade urbana que conecta passageiros, motoristas e administradores em um ecossistema digital unificado. Seu objetivo é otimizar a solicitação e realização de corridas, oferecendo uma experiência intuitiva e segura tanto para usuários quanto para motoristas, além de permitir o acompanhamento em tempo real, integração com meios de pagamento (PIX/dinheiro) e recursos administrativos para análise de métricas, gestão de regiões e resolução de disputas.

2. Modelos de Usuário e Requisitos

Esta seção tem como propósito caracterizar os usuários e atores envolvidos no sistema, além de especificar os requisitos que deverão ser atendidos. Para isso, são apresentadas descrições resumidas de cada ator, acompanhadas de modelos que representam seu papel como usuário da aplicação. Em seguida, são expostos o diagrama de casos de uso e as histórias de usuário, que servirão como base para orientar o desenvolvimento do sistema. Por fim, são disponibilizados o diagrama de sequência e o contrato de operações.

2.1 Descrição de Atores

Passageiro: Usuário que utiliza o aplicativo para solicitar corridas. É responsável por informar origem e destino, acompanhar o trajeto em tempo real, realizar o pagamento e avaliar o motorista ao final da viagem. Seu principal objetivo é realizar deslocamentos com praticidade, segurança e previsibilidade de custo.

Motorista: Usuário que utiliza o aplicativo para oferecer o serviço de transporte. Recebe solicitações de corridas, navega até o ponto de embarque e conduz o passageiro ao destino informado. Pode consultar seu histórico de corridas e acompanhar seus ganhos. É responsável por manter informações atualizadas no sistema e aceitar ou recusar corridas conforme disponibilidade.

Administrador: Responsável por gerenciar o funcionamento da plataforma por meio de um sistema web. Controla o cadastro de passageiros e motoristas, monitora as corridas realizadas, acompanha indicadores de utilização e gera relatórios financeiros. Também pode configurar taxas da plataforma e atuar em situações que demandem suporte ou resolução de conflitos.

2.2 Modelos de Usuários

Esta subseção tem como objetivo apresentar os modelos de usuários construídos a partir da definição de personas. Para a elaboração dessas personas, foram consideradas as características dos principais atores do sistema como passageiros, motoristas e administradores, bem como suas necessidades, dores e objetivos dentro do contexto da aplicação. As personas a seguir representam perfis típicos dos usuários previstos para a plataforma e servem como referência para orientar decisões de *design*, desenvolvimento e priorização de funcionalidades do sistema.

A Tabela 1 descreve a persona da usuária Mariana Silva, uma passageira que depende do transporte urbano para deslocamentos diários. É possível identificar que, apesar de estar familiarizada com aplicativos de uso comum como *WhatsApp* e *Instagram*, ela sente dificuldades em confiar nos aplicativos locais de transporte por considerá-los confusos e pouco práticos. Também é perceptível que Mariana valoriza a previsibilidade no preço e segurança durante as viagens. Por fim, após análise, observa-se que ela deseja ter maior clareza no custo antes da corrida e a possibilidade de acompanhar o trajeto em tempo real, o que lhe traria mais confiança no serviço.

Mariana Silva	
Descrição	Mariana possui 27 anos, nasceu e cresceu na cidade onde o aplicativo será implantado. Trabalha como recepcionista em uma clínica e depende de transporte diário para se deslocar ao trabalho e para compromissos pessoais. Costuma usar aplicativos como <i>WhatsApp</i> e <i>Instagram</i> , mas nunca utilizou os aplicativos de transporte já existentes na cidade, pois os considera confusos e pouco confiáveis. Busca uma alternativa prática que lhe permita se deslocar com previsibilidade de preço e segurança.
Dores	<ul style="list-style-type: none"> • Dificuldade em prever o valor de cada corrida com os aplicativos atuais. • Insegurança por não poder acompanhar a rota em tempo real. • Pouca confiança em aplicativos locais, pela falta de avaliação de motoristas.
Objetivos	<ul style="list-style-type: none"> • Ter acesso a corridas rápidas, seguras e acessíveis. • Saber o valor estimado da corrida antes de embarcar. • Acompanhar em tempo real o trajeto, sentindo-se mais segura.

Tabela 1. Persona Mariana Silva

A Tabela 2 descreve a persona do usuário Carlos Andrade, um motorista que utiliza o transporte por aplicativo como fonte de renda complementar. Identifica-se que ele já teve experiências anteriores com aplicativos locais, mas considera que eles carecem de recursos de apoio ao motorista e transparência em relação aos ganhos. Nota-se também que Carlos valoriza ferramentas que o ajudam a organizar seu trabalho de forma prática, sem exigir aprendizado complexo de novas tecnologias. Por fim, após análise, percebe-se que ele busca um sistema confiável que facilite o recebimento de corridas e forneça relatórios claros de viagens e rendimentos.

Carlos Andrade	
Descrição	Carlos tem 42 anos e trabalha como motorista de aplicativo em tempo parcial para complementar a renda da família. Já testou os aplicativos locais (LevaAli e MobyGo), mas considera que eles oferecem pouco suporte e não têm transparência nos ganhos. Ele valoriza poder organizar suas corridas e acompanhar de forma simples quanto está ganhando por dia e por semana. Carlos utiliza apenas o necessário em termos de tecnologia, mas aprende rápido quando percebe que o recurso facilita sua rotina.
Dores	<ul style="list-style-type: none"> ● Falta de relatórios claros sobre corridas e ganhos nos aplicativos atuais. ● Necessidade de depender de chamadas aleatórias, sem controle. ● Dificuldade em manter contato com suporte ou resolver problemas por falta de recursos internos no <i>app</i>.
Objetivos	<ul style="list-style-type: none"> ● Receber chamadas de corridas de forma simples e rápida. ● Acompanhar histórico de viagens e rendimento financeiro. ● Trabalhar em um sistema confiável que ofereça suporte adequado.

Tabela 2. Persona Carlos Andrade

A Tabela 3 descreve a persona da usuária Fernanda Oliveira, que atua como administradora da plataforma. É possível observar que ela precisa equilibrar a gestão do sistema com suas outras responsabilidades profissionais, o que exige relatórios padronizados e práticos para tomada de decisão. Identifica-se ainda que Fernanda encontra dificuldades em monitorar o desempenho da plataforma com os recursos limitados oferecidos pelos aplicativos concorrentes. Por fim, após análise, verifica-se que sua principal necessidade é dispor de ferramentas que permitam acompanhar a satisfação dos usuários, controlar cadastros e configurar taxas de forma objetiva.

Fernanda Oliveira	
Descrição	Fernanda possui 35 anos, formada em Administração e atua como secretária de gestor da plataforma. Seu papel é garantir que motoristas e passageiros estejam devidamente cadastrados e que o sistema funcione de forma estável. Ela precisa ter acesso a relatórios que mostrem número de corridas realizadas, ganhos, taxas e indicadores de uso do sistema, de modo a tomar decisões sobre melhorias e acompanhar o crescimento do aplicativo.
Dores	<ul style="list-style-type: none"> ● Falta de relatórios padronizados nos aplicativos locais. ● Dificuldade em monitorar qualidade do serviço (avaliações e <i>feedbacks</i>). ● Necessidade de ajustar taxas e políticas de forma manual.
Objetivos	<ul style="list-style-type: none"> ● Ter acesso a relatórios claros e detalhados sobre o funcionamento do sistema. ● Acompanhar o desempenho de motoristas e satisfação de passageiros. ● Configurar taxas e gerenciar usuários de forma prática.

Tabela 3. Persona Fernanda Oliveira

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta subseção tem como finalidade apresentar os casos de uso e as histórias de usuário que orientam o desenvolvimento do sistema. Para isso, é exibido o diagrama de casos de uso, no qual estão representadas as principais interações entre os atores e a aplicação. Em complemento, são descritas as histórias de usuário correspondentes, detalhando as funcionalidades esperadas e permitindo uma visão clara dos requisitos que devem ser implementados.

2.3.1 Diagrama de Casos de Uso

A Figura 1 apresenta o diagrama de casos de uso do sistema proposto. No diagrama, são representados três atores principais: Passageiro, Motorista e Administrador, que interagem diretamente com o sistema. Além deles, são exibidos dois sistemas externos: o Serviço de Mapas e Geolocalização, utilizado para cálculo de rotas e estimativas de tempo, e o *Gateway* de Pagamentos(Futuramente). O passageiro acessa o sistema para criar conta, solicitar corridas, acompanhar o trajeto, efetuar pagamento e avaliar o motorista. O Motorista utiliza o sistema para receber corridas, aceitar ou recusar solicitações, navegar até o passageiro e ao destino, marcar etapas da viagem e consultar relatórios de ganhos. O Administrador gerencia usuários, motoristas, taxas, regiões de operação, relatórios do sistema e solicitações de suporte. Ela representa visualmente essas interações, bem como as relações entre os casos de uso e os sistemas externos.

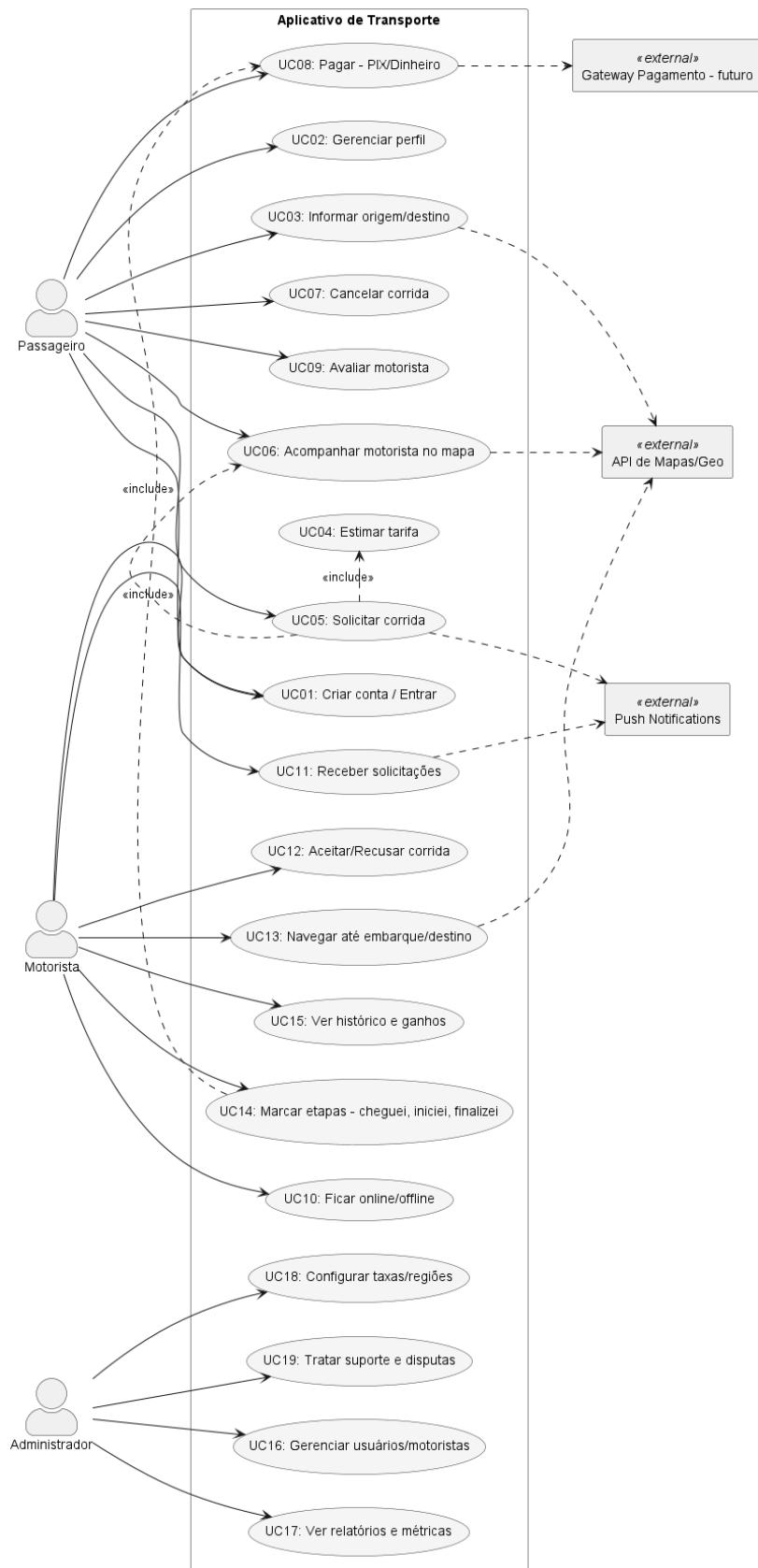


Figura 1 – Diagrama de Casos de Uso

2.3.2 Histórias de Usuário

Nesta seção, são listadas as histórias de usuário levantadas para o sistema proposto. Para fins de organização, utiliza-se identificadores no formato *US#ID*, em que *US* se refere a *User Story*.

As histórias de usuário identificadas para o sistema são:

- US1. Como passageiro, eu gostaria de criar uma conta e gerenciar meu perfil para que eu possa acessar o sistema e utilizar seus serviços.
- US2. Como passageiro, eu gostaria de informar origem e destino para que eu possa solicitar uma corrida.
- US3. Como passageiro, eu gostaria de ver a estimativa de tarifa e tempo de chegada do motorista para decidir se quero confirmar a corrida.
- US4. Como passageiro, eu gostaria de acompanhar a rota do motorista em tempo real para saber onde ele está.
- US5. Como passageiro, eu gostaria de efetuar o pagamento via PIX ou em dinheiro para concluir a corrida com segurança.
- US6. Como passageiro, eu gostaria de avaliar o motorista após a corrida para contribuir com a reputação da plataforma.
- US7. Como motorista, eu gostaria de entrar no sistema e definir meu *status* como *online* ou *offline* para indicar quando estou disponível para receber corridas.
- US8. Como motorista, eu gostaria de receber solicitações de corrida com informações de origem, destino e valor estimado para decidir se aceito a viagem.
- US9. Como motorista, eu gostaria de navegar até o passageiro e depois até o destino usando o mapa do *app* para otimizar a rota.
- US10. Como motorista, eu gostaria de marcar as etapas da corrida (cheguei, iniciei, finalizei) para registrar corretamente o andamento da viagem.
- US11. Como motorista, eu gostaria de acessar um painel com meus ganhos diários e semanais para acompanhar meus resultados.
- US12. Como administrador, eu gostaria de gerenciar contas de passageiros e motoristas para manter o sistema organizado e seguro.

US13. Como administrador, eu gostaria de acompanhar relatórios de corridas e métricas do sistema para tomar decisões estratégicas.

US14. Como administrador, eu gostaria de configurar taxas e regiões de operação para ajustar o funcionamento da plataforma conforme a cidade.

US15. Como administrador, eu gostaria de visualizar e responder solicitações de suporte ou disputas para resolver problemas de forma rápida.

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Nesta seção, são apresentados os diagramas de sequência do sistema, assim como seus Contratos de Operações. O objetivo destes diagramas é descrever os fluxos de interação existentes entre os usuários e o sistema, representando a troca de mensagens e as dependências entre módulos da aplicação.

A Figura 2 representa o diagrama de sequência do sistema relacionado ao fluxo de solicitação de corrida. Nesse fluxo, o Passageiro informa sua origem e destino e, ao confirmar o pedido, o sistema consulta o Serviço de Mapas para calcular a rota e o tempo estimado.

Após essa etapa, a aplicação envia uma notificação em tempo real ao Motorista disponível mais próximo, que poderá aceitar ou recusar a corrida.

Esse diagrama está diretamente relacionado aos casos de uso UC03 (Informar origem/destino), UC04 (Estimar tarifa), UC05 (Solicitar corrida) e UC11 (Receber solicitação).

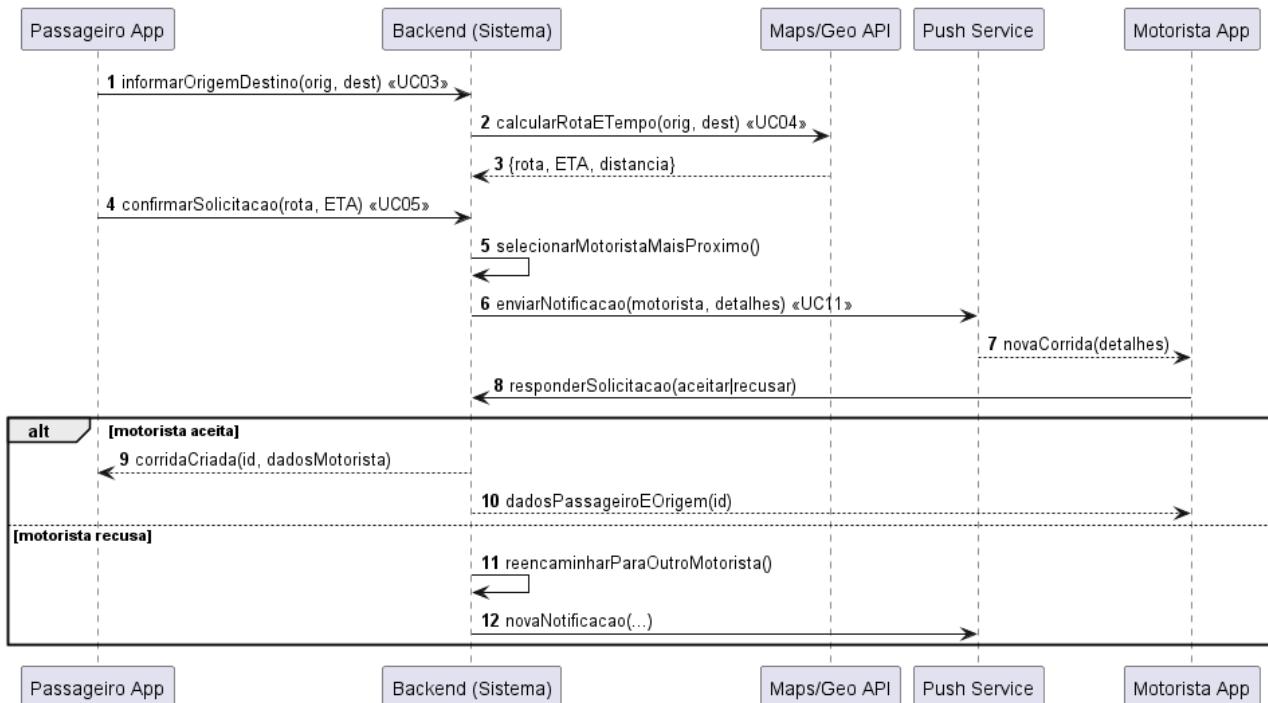


Figura 2 – Diagrama de Sequência Solicitar Corrida e Notificar Motorista

Contrato	Solicitar corrida e notificar motorista
Operação	<i>requestRide(origin: String, destination: String)</i>
Referências cruzadas	UC03 Informar origem/destino, UC04 Estimar tarifa, UC05 Solicitar corrida, UC11 Receber solicitação
Pré-condições	O passageiro deve estar autenticado no sistema e com geolocalização ativa
Pós-condições	O motorista mais próximo é notificado e a corrida é registrada no banco de dados como “pendente”

A Figura 3 mostra o diagrama de sequência referente ao fluxo de pagamento da corrida. Após o término da viagem, o sistema apresenta as opções de PIX ou Dinheiro. No processo de pagamento em dinheiro ou pix, o sistema atualiza o status localmente. Esse fluxo está relacionado aos casos de uso UC08 (Efetuar pagamento) e UC14 (Finalizar corrida).

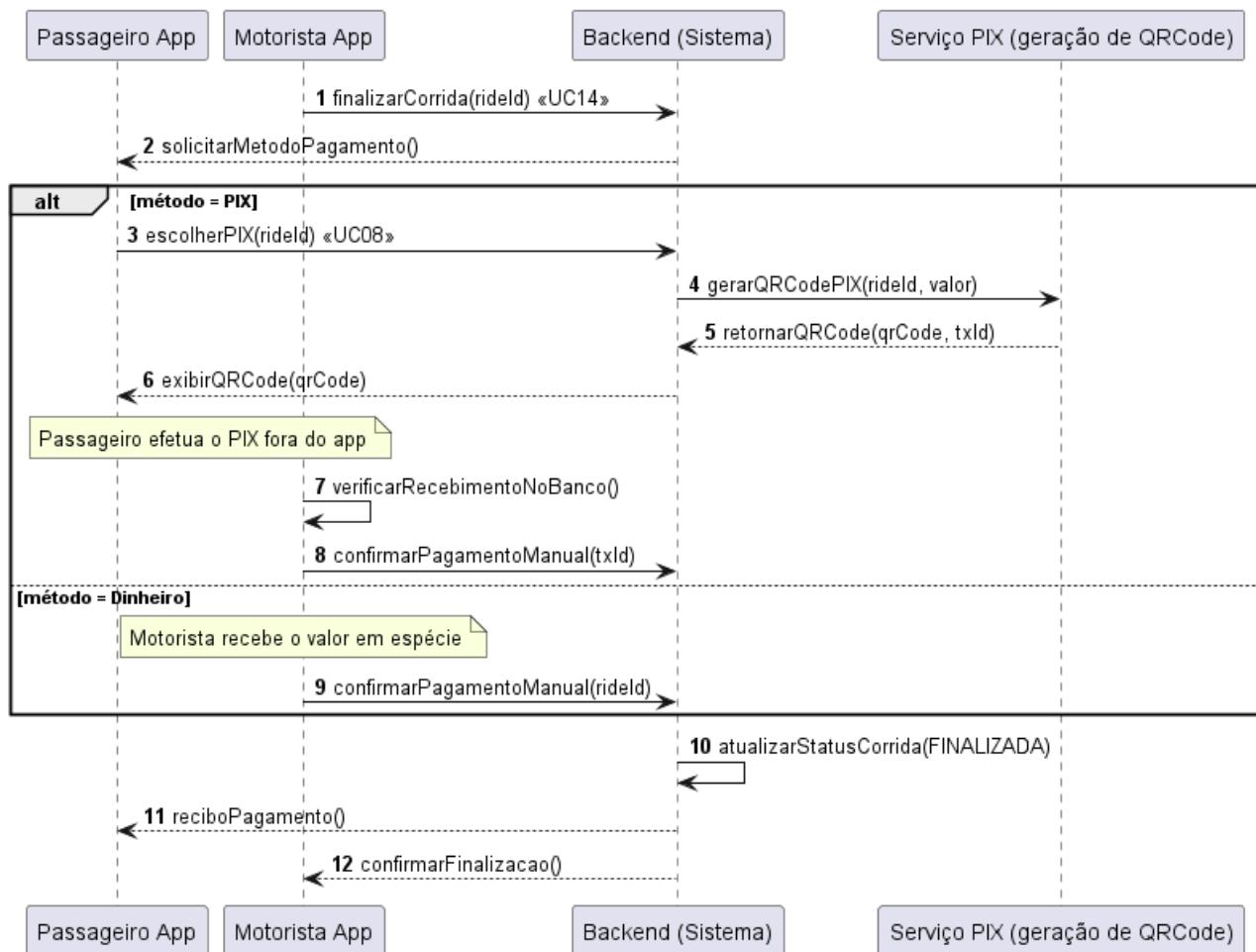


Figura 3 – Diagrama de Sequência Efetuar Pagamento

Contrato	Efetuar pagamento da corrida
Operação	<i>processPayment(paymentMethod: Enum, rideId: Int)</i>
Referências cruzadas	UC08 Efetuar pagamento, UC14 Finalizar corrida

Pré-condições	A corrida deve ter sido concluída e estar aguardando pagamento
Pós-condições	O pagamento é confirmado (PIX) ou marcado como concluído (dinheiro), alterando o status da corrida para “finalizada”

A Figura 4 representa o diagrama de sequência do fluxo de avaliação do motorista. Após o encerramento da corrida e o processamento do pagamento, o Passageiro pode registrar uma avaliação que inclui nota e comentário. O sistema valida os dados e salva a avaliação, atualizando a média do motorista no banco de dados. Esse fluxo está associado ao caso de uso UC09 (Avaliar motorista).

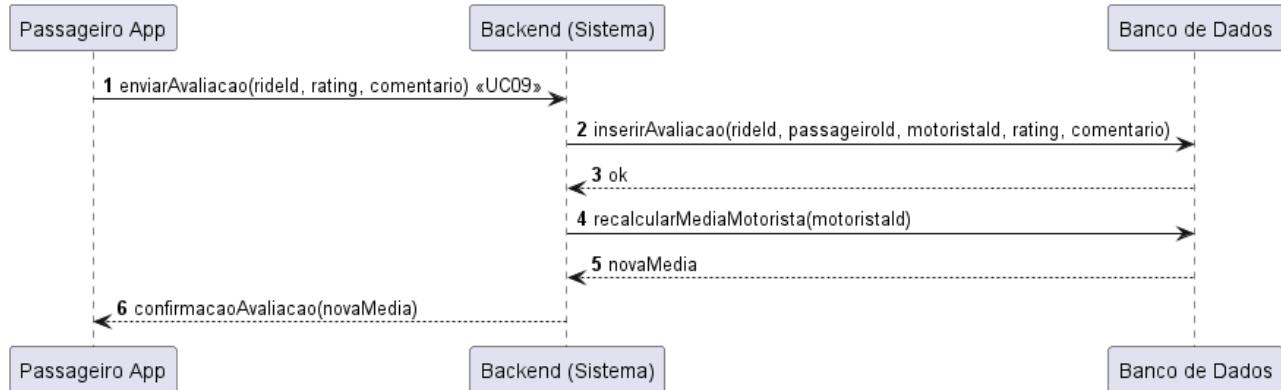


Figura 4 – Diagrama de Sequência Avaliação do Motorista

Contrato	Registrar avaliação do motorista
Operação	<code>submitReview(rideId: Int, rating: Int, comment: String)</code>
Referências cruzadas	UC09 Avaliar motorista
Pré-condições	A corrida deve ter sido finalizada e o pagamento concluído
Pós-condições	A avaliação é registrada e a nota média do motorista é atualizada no sistema

A Figura 5 ilustra o diagrama de sequência do painel administrativo, que demonstra como o Administrador acessa o sistema web, solicita relatórios e recebe informações agregadas de corridas, motoristas e usuários. O fluxo inclui a requisição dos dados, o processamento interno e a resposta exibida em forma de gráficos e métricas. Esse diagrama se relaciona aos casos de uso UC16 (Gerenciar usuários/motoristas), UC17 (Ver relatórios e métricas) e UC18 (Configurar taxas/regiones).

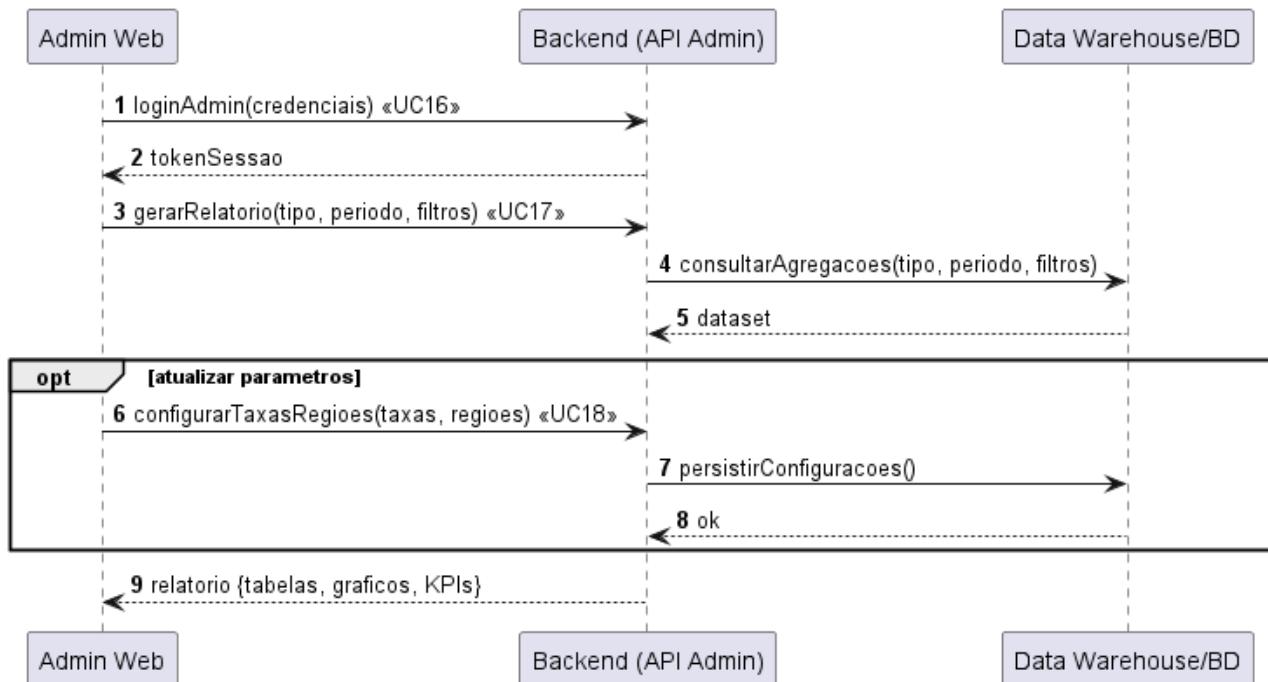


Figura 5 – Diagrama de Sequência Gerar Relatórios Administrativos

Contrato	Gerar relatórios administrativos
Operação	<code>generateReport(type: Enum, dateRange: String)</code>
Referências cruzadas	UC16 Gerenciar usuários, UC17 Ver relatórios, UC18 Configurar taxas
Pré-condições	O administrador deve estar autenticado no painel web
Pós-condições	O relatório solicitado é gerado e apresentado na interface em formato gráfico

3. Modelos de Projeto

Nesta seção é apresentada a modelagem dos objetos e fluxos que compõem o sistema proposto. Para isso, foram desenvolvidos diagramas de classes que descrevem os principais pacotes do sistema: *controllers*, *services*, *domain/models*, *repositories* e *dtos*. Cada um deles tem como objetivo representar as responsabilidades e inter-relações entre os componentes que compõem o aplicativo de transporte.

Os diagramas a seguir foram projetados para fornecer uma visão clara e modular da arquitetura, evidenciando como os diferentes níveis — interface, lógica de negócios, persistência e dados — interagem para garantir o funcionamento adequado do sistema.

3.1 Diagrama de Classes

A seguir é apresentado o diagrama de classes referente ao pacote de *controllers* do sistema. Essas classes têm como responsabilidade intermediar as requisições oriundas do aplicativo móvel com as camadas de serviço, realizando a comunicação por meio de uma *API REST*.

O pacote contém controladores para autenticação, operações de passageiros, operações de motoristas, gerenciamento de corridas, pagamentos e administração.

Cada controlador expõe métodos que representam os principais casos de uso da aplicação, como solicitar corrida, responder a uma solicitação, acompanhar viagens, registrar pagamentos e gerar relatórios administrativos.

A Figura 6 ilustra o diagrama de classes do pacote *controllers*, evidenciando as dependências diretas entre os controladores e os serviços correspondentes que contêm as regras de negócio.

É possível observar a relação entre *PassageiroController* e *CorridaService*, *MotoristaController* e *RelatorioService*, entre outras.

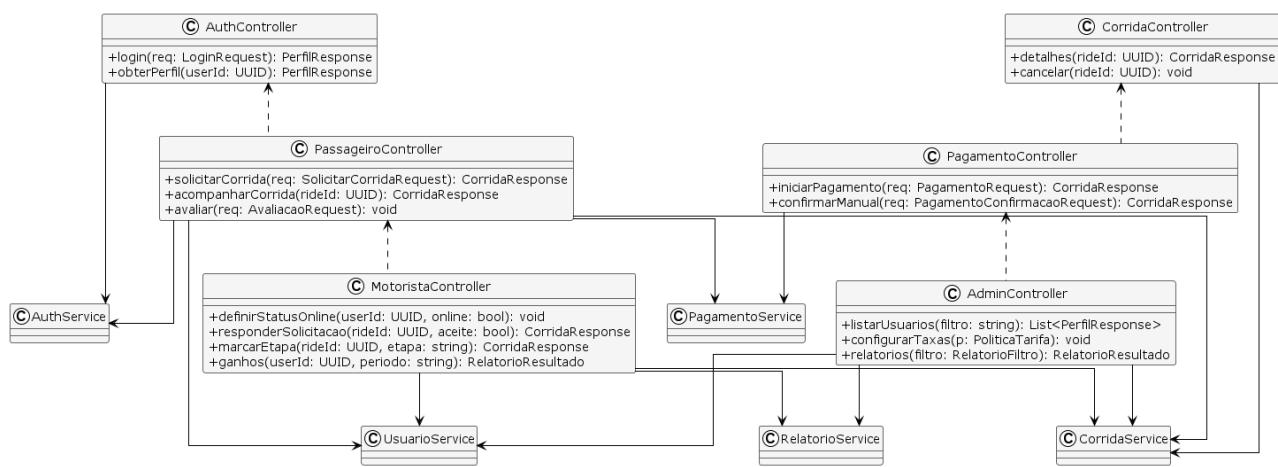


Figura 6 – Diagrama de Classes: Controllers

O diagrama de classes apresentado a seguir representa o pacote de *services*, responsável pela camada de regras de negócio do aplicativo.

Essas classes implementam as operações centrais da aplicação, como o gerenciamento de corridas, cálculo de tarifas, pareamento entre motoristas e passageiros, processamento de pagamentos e geração de relatórios.

O pacote inclui serviços de autenticação (*AuthService*), gestão de usuários (*UsuarioService*), cálculo de preço dinâmico (*PrecificacaoService*), correspondência de motoristas (*MatchingService*), gerenciamento de *status* (*MotoristaStatusService*), controle de corridas (*CorridaService*), pagamentos (*PagamentoService*) e notificações (*NotificacaoService*).

Há ainda o *RelatorioService*, que fornece métricas de desempenho e estatísticas operacionais tanto para motoristas quanto para administradores.

As Figuras 7 e 8 apresentam o diagrama de classes dos *services*, mostrando as dependências com os repositórios de persistência e com integrações externas como *APIs* de mapas e serviços de *push*.

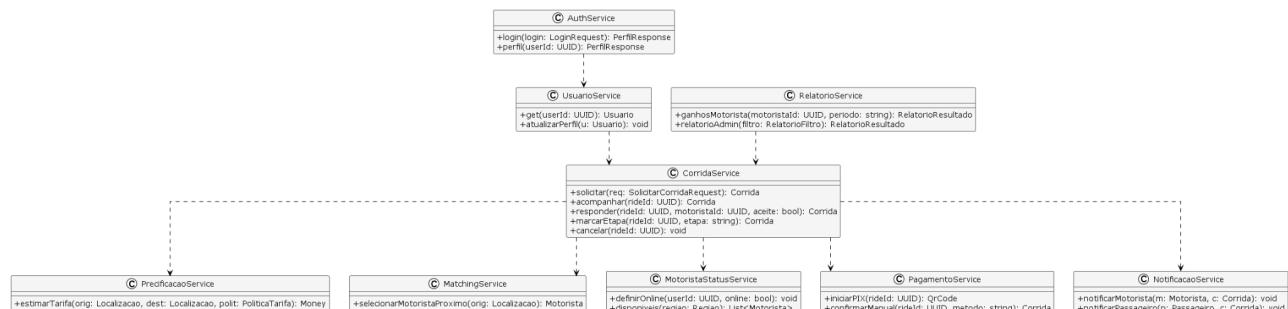


Figura 7 – Diagrama de Classes: Services01

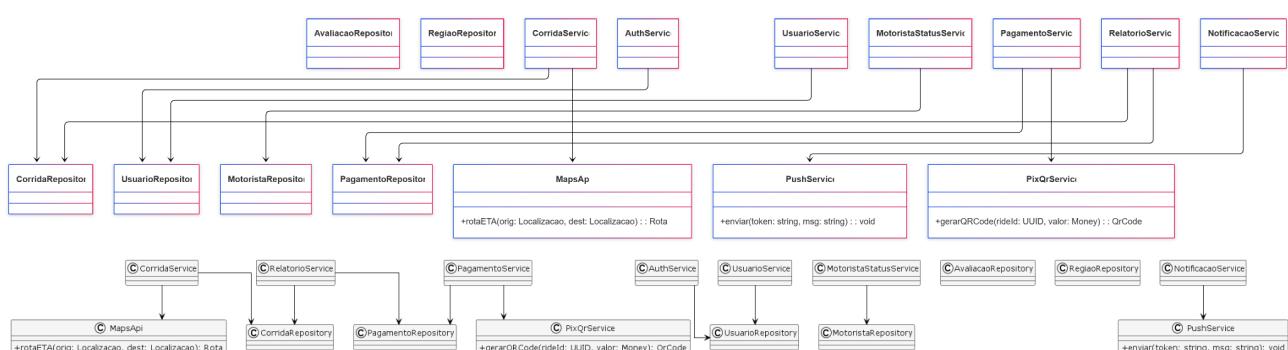


Figura 8 – Diagrama de Classes: Services02

O diagrama de classes a seguir descreve o pacote *domain*, que contém os modelos de dados fundamentais do sistema. Essas classes representam as entidades centrais que sustentam o funcionamento do aplicativo, como *Usuario*, *Passageiro*, *Motorista*, *Veiculo*, *Corrida*, *Pagamento* e *Avaliacao*.

Cada entidade foi estruturada com seus respectivos atributos e relacionamentos, garantindo coerência com os processos do negócio. Por exemplo, a classe Corrida mantém vínculos diretos com Passageiro, Motorista e Rota, além de conter informações de status, valores e horários. A classe Pagamento está associada à Corrida, representando tanto pagamentos via PIX quanto em dinheiro, ambos processados manualmente no aplicativo. Além disso, o modelo inclui entidades auxiliares como Localizacao, PoliticaTarifa, Regiao e Money, usadas para padronizar endereços, políticas de preço e valores monetários.

A Figura 9 apresenta o diagrama de classes do domínio, permitindo visualizar de forma clara as relações entre as entidades e seus papéis dentro do fluxo geral da aplicação.

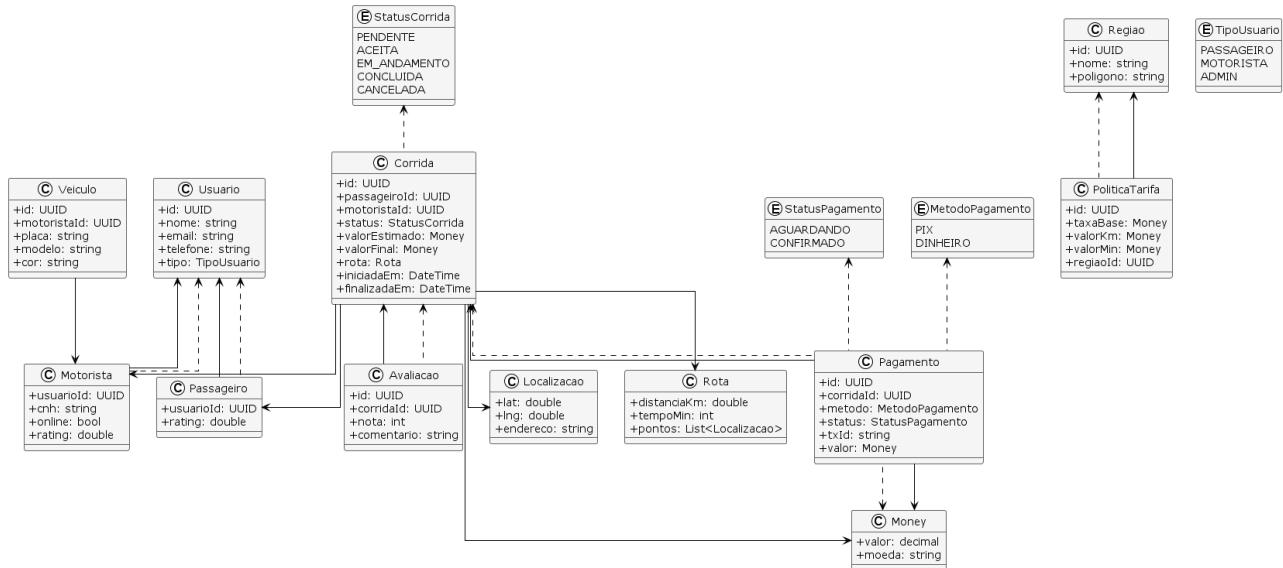


Figura 9 – Diagrama de Classes: Domain / Models

O diagrama de classes a seguir representa o pacote de *repositories*, responsável pela camada de persistência de dados. Essas classes são definidas como interfaces, pois a implementação pode variar de acordo com a tecnologia de banco de dados adotada, mantendo assim o princípio da abstração e separação de responsabilidades.

Cada repositório é responsável por manipular uma entidade específica, garantindo a persistência e recuperação de informações essenciais ao funcionamento do sistema. Entre os repositórios estão *UsuarioRepository*, *CorridaRepository*, *PagamentoRepository* e *AvaliacaoRepository*, que lidam com as operações de CRUD e consultas filtradas.

A Figura 10 exibe o diagrama de classes de *repositories*, demonstrando a estrutura modular de acesso aos dados e sua conexão com a camada de *services*.

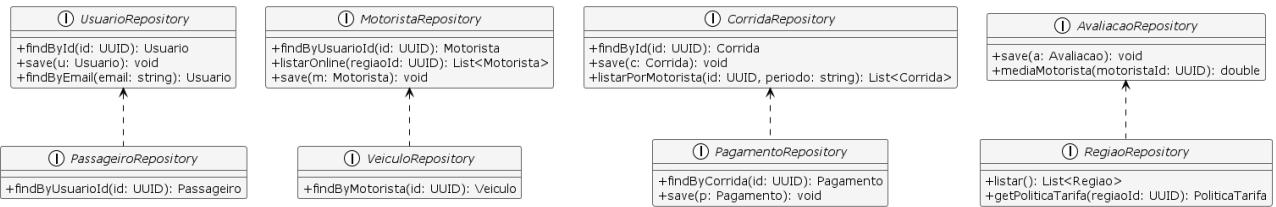


Figura 10 – Diagrama de Classes: Repositories

Por fim, a figura a seguir apresenta o diagrama de classes referente aos *Data Transfer Objects* (*DTOs*), que têm como função padronizar a comunicação entre o cliente (aplicativo móvel) e o servidor (API). Esses objetos simplificam o envio e recebimento de dados, evitando o tráfego de entidades completas e protegendo a estrutura interna do sistema.

Entre os principais *DTOs* estão `LoginRequest`, `PerfilResponse`, `SolicitarCorridaRequest`, `CorridaResponse`, `PagamentoRequest`, `PagamentoConfirmacaoRequest` e `AvaliacaoRequest`. Também foram definidos objetos voltados à geração de relatórios, como `RelatorioFiltro` e `RelatorioResultado`, utilizados por administradores e motoristas para visualizar métricas de desempenho.

A Figura 11 apresenta o diagrama de classes dos *DTOs*, mostrando a utilização de tipos de domínio como `Localizacao`, `Rota` e `Money` para garantir consistência entre os dados de transporte e pagamento.

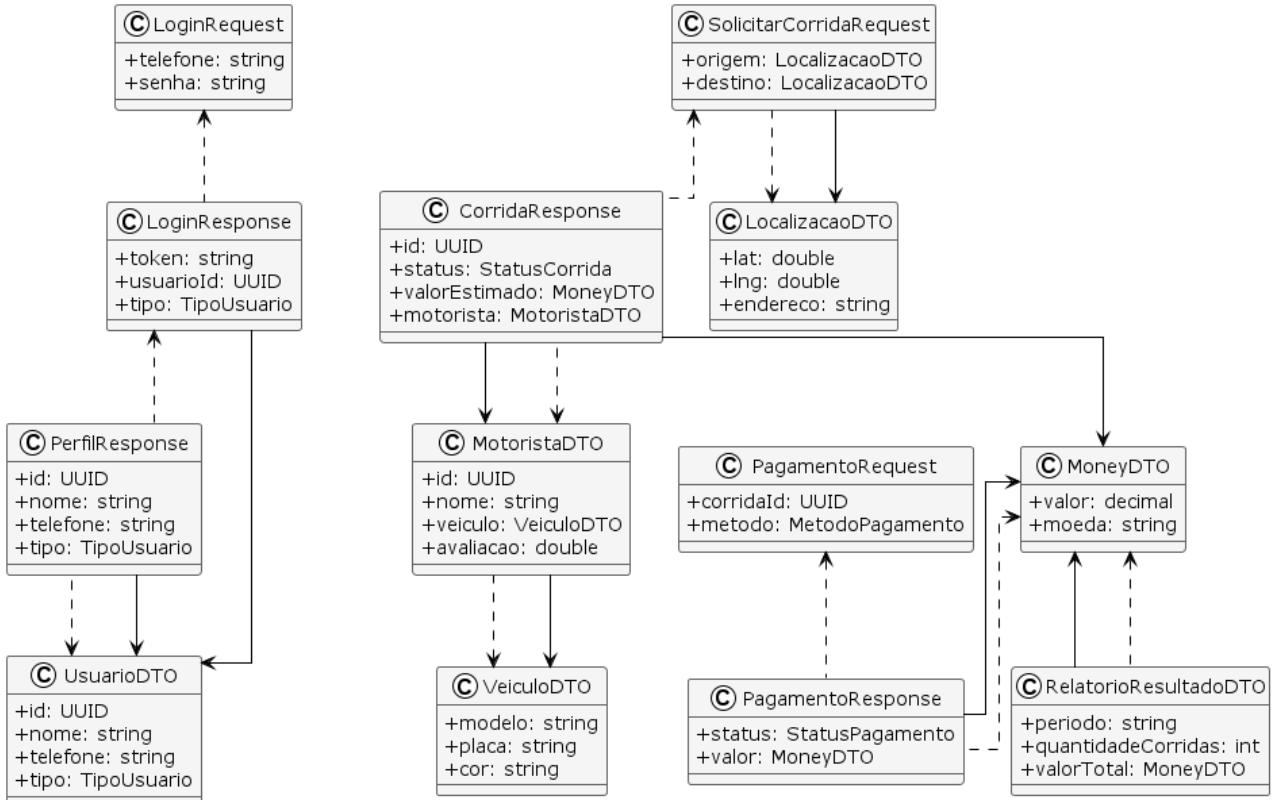


Figura 11 – Diagrama de Classes: DTOs

3.2 Diagramas de Sequência

Nesta seção, são apresentados os diagramas de sequência que modelam os fluxos principais do sistema de transporte proposto. Esses diagramas têm como objetivo mapear os caminhos críticos de uso da aplicação, evidenciando quem se comunica com quem, quais mensagens são trocadas e em que ordem.

Foram selecionados quatro fluxos representativos: solicitação de corrida, pagamento (PIX/dinheiro, confirmação manual), avaliação do motorista e relatórios administrativos. Juntos, eles cobrem as interações essenciais entre Passageiro, Motorista e Administrador, além das integrações necessárias com serviços externos.

A Figura 12 representa o fluxo de solicitação de corrida. O Passageiro informa origem e destino; o sistema consulta o serviço de mapas para obter rota, distância e tempo estimado; em seguida, identifica o motorista mais próximo e envia uma notificação de corrida. O motorista pode aceitar ou recusar. Em caso de recusa, a solicitação é reencaminhada a outro motorista. Este fluxo está relacionado aos casos de uso UC03 (Informar origem/destino), UC04 (Estimar tarifa), UC05 (Solicitar corrida) e UC11 (Receber solicitação).

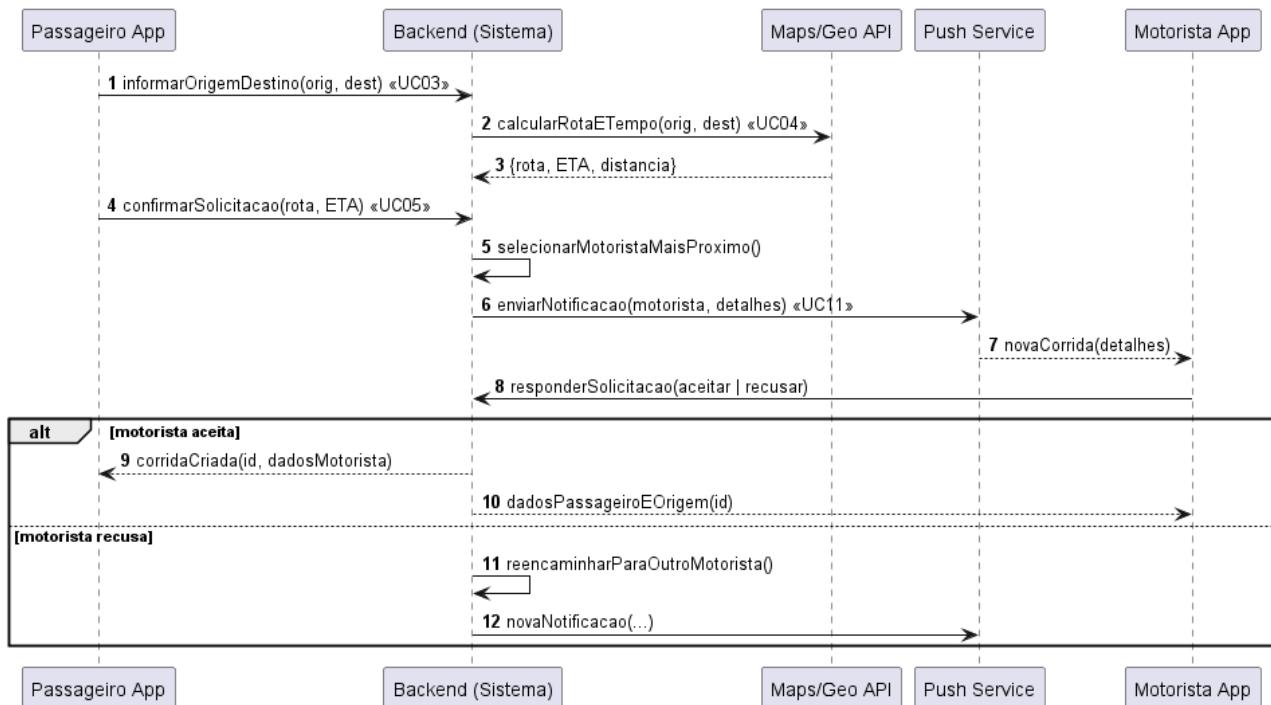


Figura 12 – Diagrama de Sequência Solicitação de Corrida

A Figura 13 descreve o encerramento da corrida e o pagamento. O sistema apresenta os métodos PIX e Dinheiro. Para PIX, o *backend* gera um *QR Code* (sem integração bancária automática); o motorista verifica manualmente no app do banco se o valor entrou e confirma no sistema. Para Dinheiro, o motorista confirma manualmente o recebimento. Em ambos os casos, a corrida passa a Finalizada, e recibos são disponibilizados. Este fluxo cobre UC08 (Efetuar pagamento) e UC14 (Finalizar corrida).

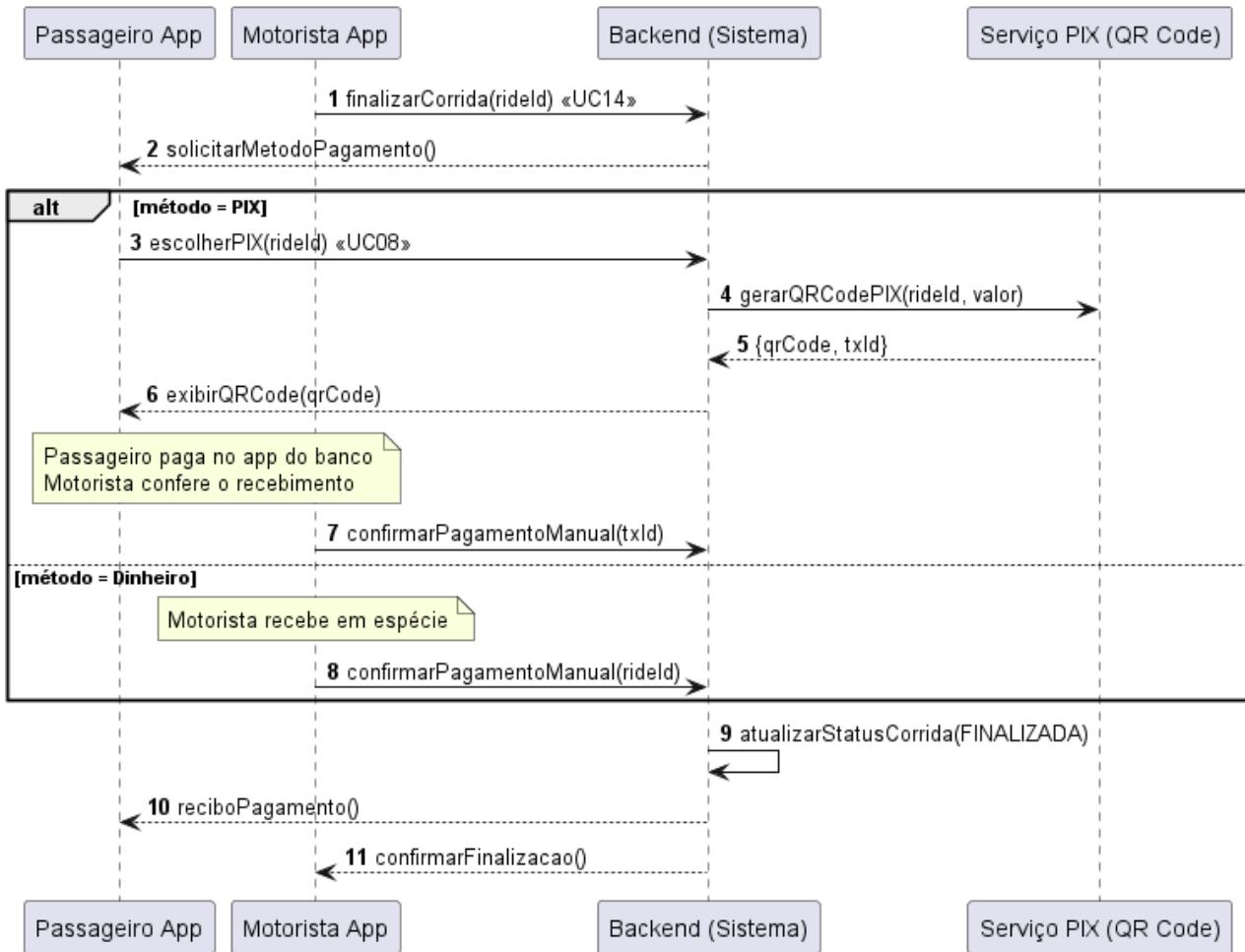


Figura 13 – Diagrama de Sequência Encerramento da Corrida e Pagamento

A Figura 14 mostra o fluxo de avaliação do motorista pelo Passageiro após o pagamento. O cliente informa nota e comentário; o sistema armazena a avaliação e recalcula a média do motorista. Esse processo retroalimenta a qualidade do serviço e influencia futuras seleções de motoristas. Relaciona-se ao caso de uso UC09 (Avaliar motorista).

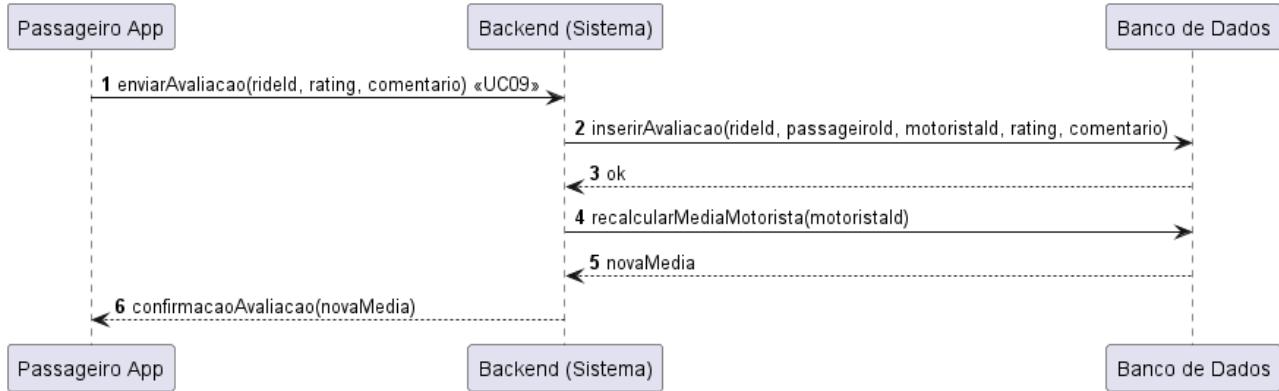


Figura 14 – Diagrama de Sequência Avaliação do Motorista

A Figura 15 ilustra o painel administrativo. O Administrador realiza login, solicita relatórios (por período, região, motorista etc.), e o *backend* consulta o repositório/warehouse para agregar dados de corridas e pagamentos. Opcionalmente, o Administrador pode ajustar configurações (taxas, regiões), que são persistidas para uso nos cálculos. Relaciona-se aos casos de uso UC16 (Gerenciar usuários/motoristas), UC17 (Ver relatórios e métricas) e UC18 (Configurar taxas/regiões).

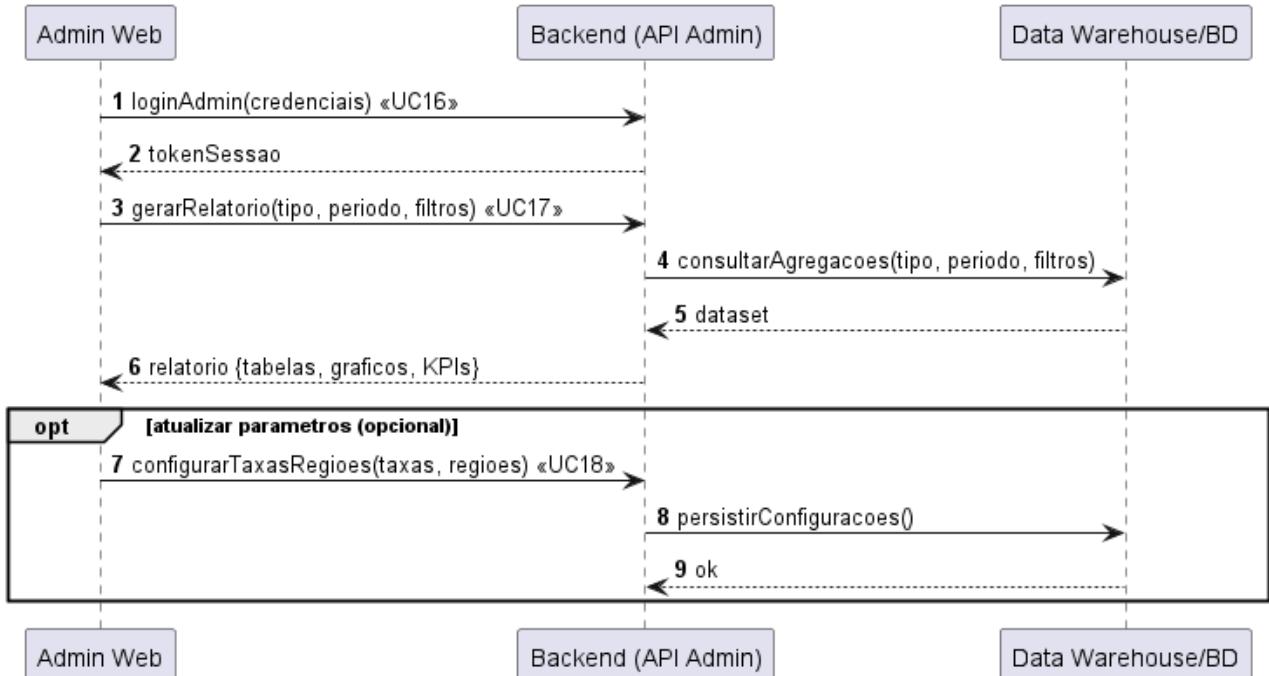


Figura 15 – Diagrama de Sequência Painel Administrativo

3.3 Diagramas de Comunicação

Nesta seção, são apresentados os diagramas de comunicação que modelam as trocas de mensagens entre atores, camadas e serviços do sistema de transporte. O objetivo é documentar quem se comunica com quem e em que ordem, cobrindo os principais fluxos: autenticação, solicitação e

despacho de corrida, atualização de *status*, pagamento e relatórios administrativos. Os diagramas a seguir complementam os diagramas de sequência, servindo como registro das interfaces e contratos de mensagem utilizados na aplicação.

A Figura 16 apresenta as trocas de mensagens para autenticação de usuários (passageiros e motoristas). O aplicativo envia as credenciais para a *API*, que delega a validação ao *AuthService*. Em seguida, os dados do usuário são obtidos no *UsuarioRepository* e um *token* de sessão é retornado ao cliente. Esse fluxo garante a segurança de acesso e inicializa o contexto de sessão para as demais operações do sistema.

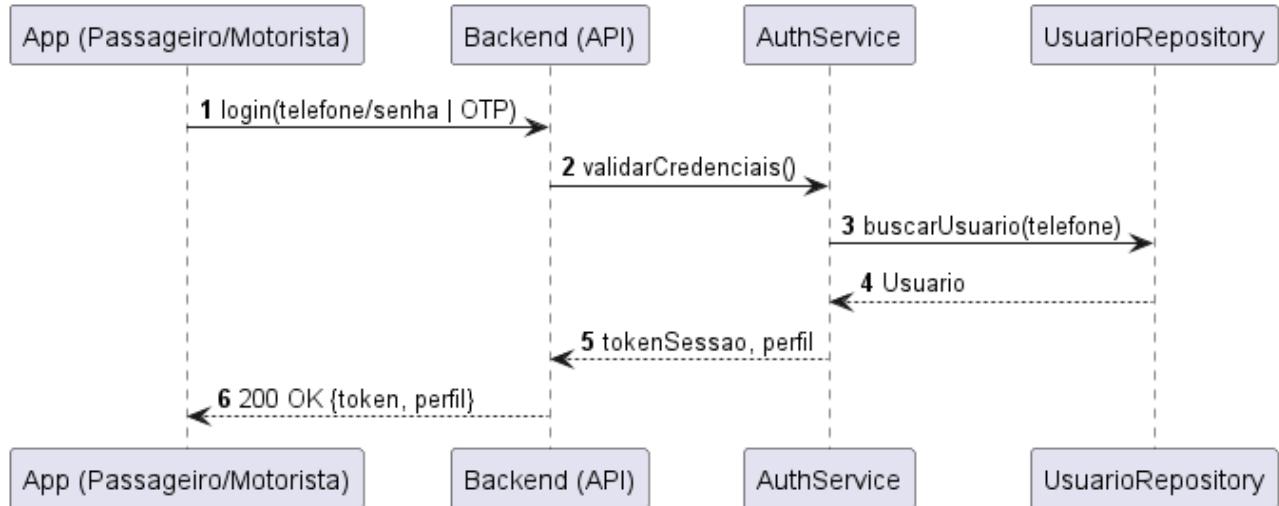


Figura 16 – Diagrama de Comunicação Autenticar Usuário

A Figura 17 descreve o fluxo de comunicação para solicitação de corrida. Após informar origem/destino, a *API* consulta a *Maps/Geo API* para estimativa de rota/tempo, chama o *MatchingService* para selecionar o motorista disponível mais próximo e utiliza o *PushService* para notificar o motorista. A resposta do motorista (aceite/recusa) retorna à *API*, que confirma ao passageiro e publica o estado inicial da corrida.

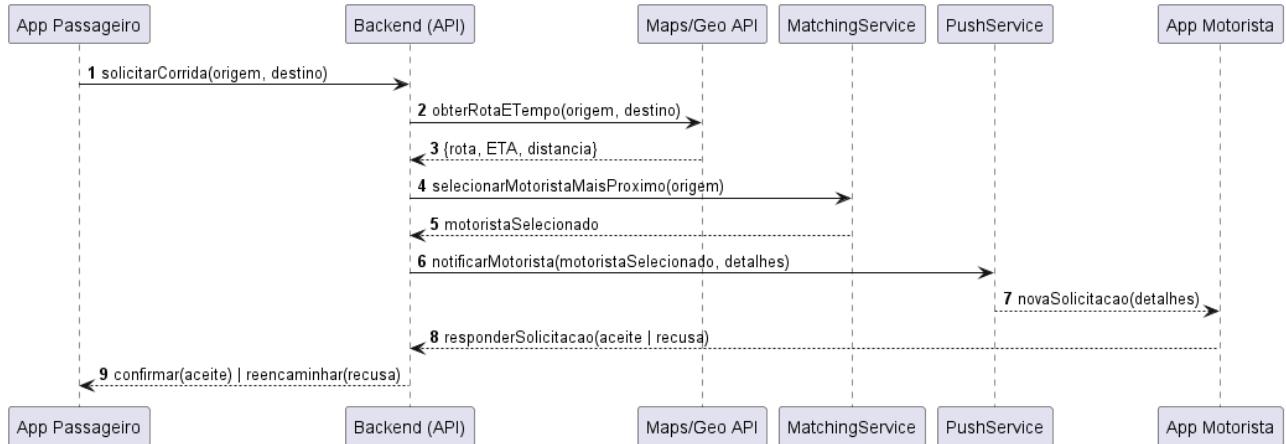


Figura 17 – Diagrama de Comunicação Solicitar Corrida e Despachar Motorista

A Figura 18 mostra as mensagens trocadas durante a execução da corrida. O motorista envia eventos de *status* (a caminho, chegou, iniciou, finalizou) à *API*. A *API* persiste no *CorridaRepository*, recalcula *ETA* quando necessário via *Maps/Geo API* e notifica passageiro/motorista com atualizações relevantes. Esse fluxo mantém ambas as partes informadas e cria o histórico operacional da corrida.

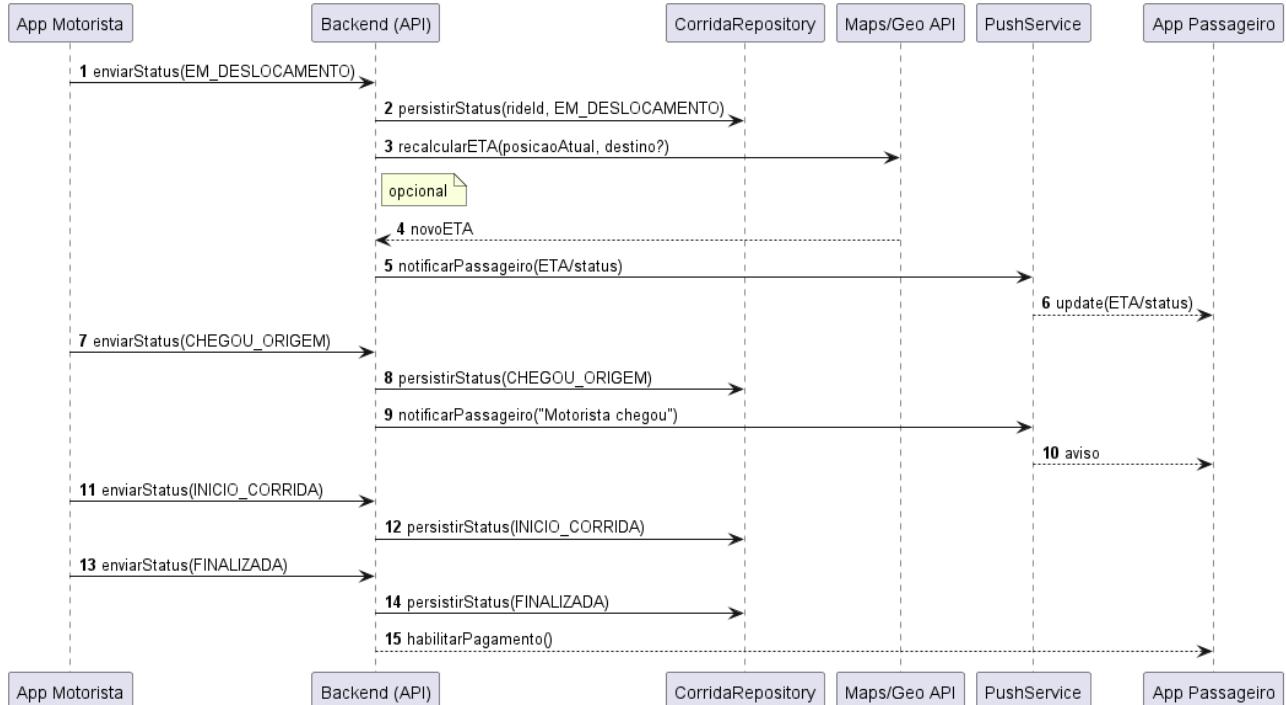


Figura 18 – Diagrama de Comunicação Atualizar Status da Corrida e Notificar Partes

A Figura 19 detalha as comunicações no pagamento. Ao finalizar a corrida, o passageiro escolhe PIX ou dinheiro. Para PIX, a *API* solicita ao *PixQrService* a geração do *QR Code* e o exibe no app; o motorista confere manualmente no seu banco e confirma na *API*. Para dinheiro, o motorista

apenas confirma o recebimento. Em ambos os casos, a *API* atualiza a Corrida para “Finalizada” e emite recibos.

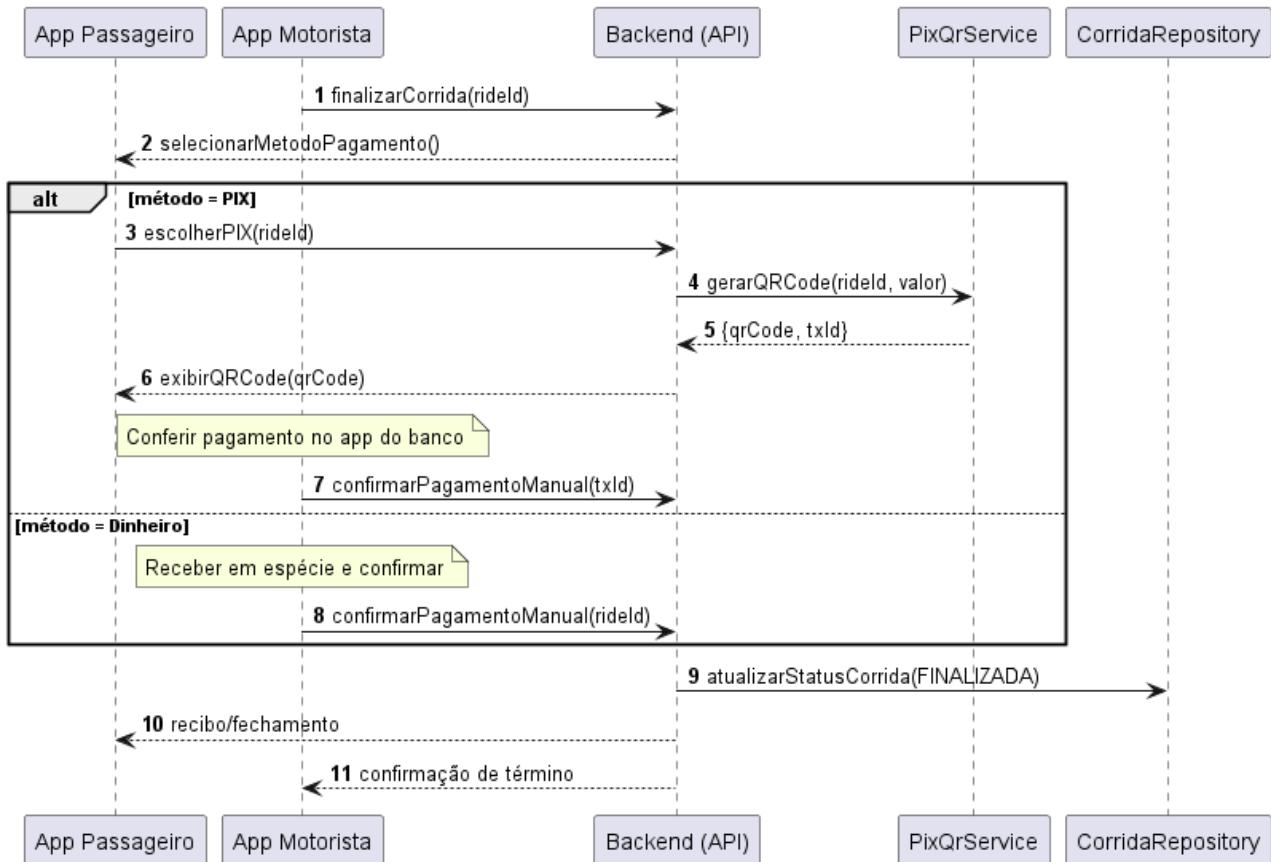


Figura 19 – Diagrama de Comunicação Efetuar Pagamento

A Figura 20 apresenta a comunicação do painel administrativo para geração de relatórios. O administrador autentica-se, informa período e filtros, e a *API* consulta o *Data Warehouse/BD* para produzir tabelas e indicadores (corridas realizadas, receita, cancelamentos, ganho por motorista/região). Opcionalmente, parâmetros (taxas/regiões) podem ser atualizados e persistidos.

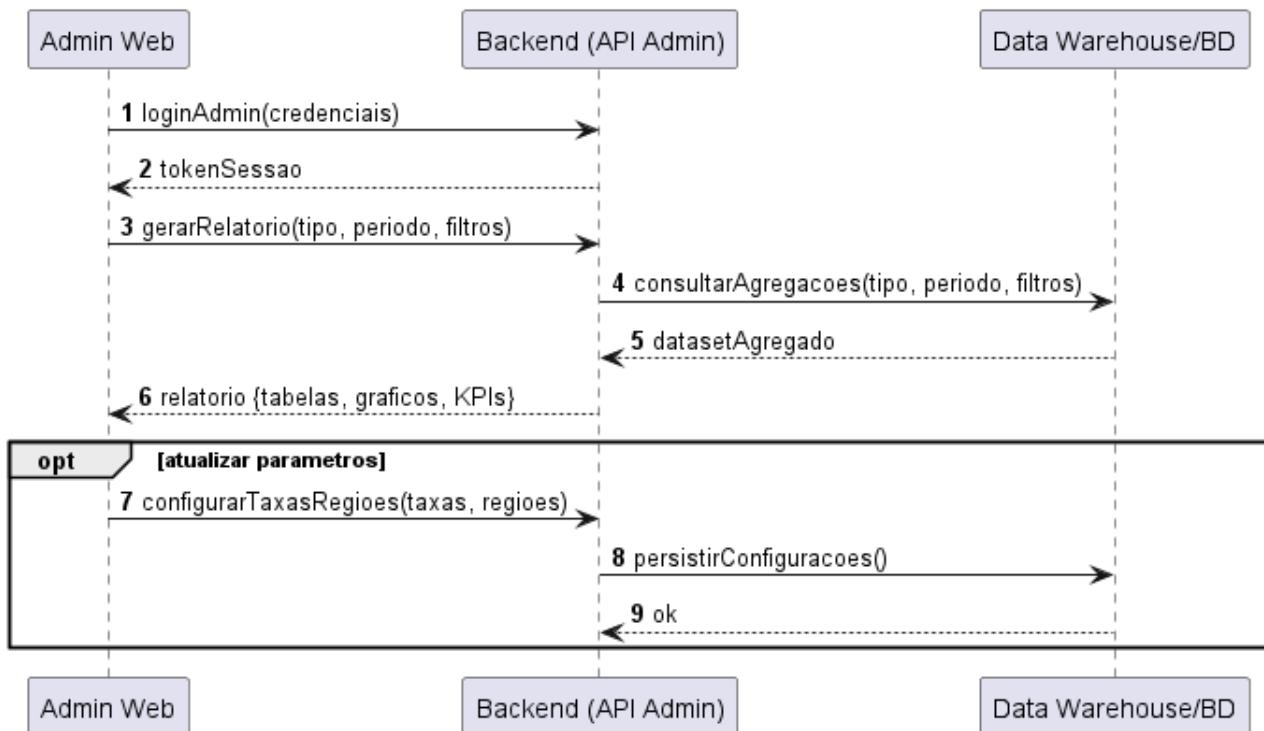


Figura 20 – Diagrama de Comunicação Gerar Relatórios Administrativos

3.4 Arquitetura

Pode ser descrita com um diagrama apropriado da UML ou C4 Model

3.5 Diagramas de Estados

Diagramas de estados do sistema.

3.6 Diagrama de Componentes e Implantação.

Diagramas de componentes do sistema. Diagrama de implantação mostrando onde os componentes estarão alocados para a execução.

4. Projeto de Interface com Usuário

Esta seção tem como objetivo apresentar e descrever as interfaces de interação com o usuário que compõem o sistema MobU. Para isso, foi elaborado um protótipo de baixa fidelidade, desenvolvido com o intuito de representar a estrutura visual e o fluxo principal de navegação da aplicação. As telas foram construídas de forma simplificada, com foco na organização dos elementos e na disposição das funcionalidades, possibilitando a validação inicial da proposta de interface antes da etapa de design detalhado. Dessa forma, as interfaces aqui apresentadas estão relacionadas aos casos de uso descritos na Seção 2.3.1, servindo como base para o mapeamento das funcionalidades essenciais que atenderão aos requisitos funcionais e não funcionais do sistema.

O objetivo deste protótipo é demonstrar a interação prevista entre os usuários e o sistema, permitindo visualizar o comportamento esperado das principais telas e apoiar o planejamento da fase de implementação.

4.1 Esboço das Interfaces Comuns a Todos os Atores

Wireframe/mockup/storyboard das interfaces que são comuns a todos os atores do sistema.

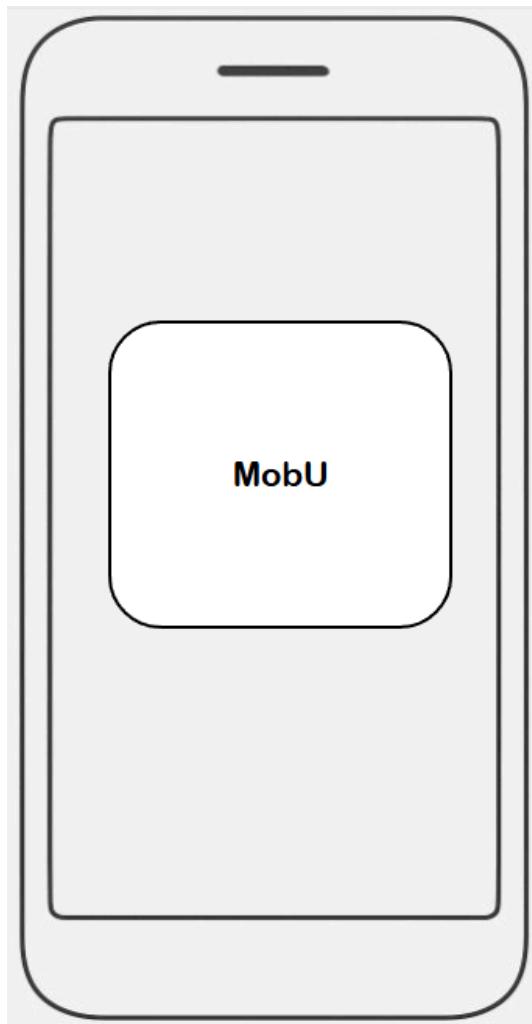


Figura 21. Página Splash Screen.



Figura 22. Página de login/cadastro.

A Figura 21 representa a tela de Splash Screen do aplicativo do passageiro. Esta página é exibida brevemente ao abrir o app e enquanto o sistema carrega dados de sessão ou verifica se o usuário já está autenticado. Caso a autenticação seja válida, a aplicação redireciona automaticamente o usuário para a página principal de solicitação de corrida (Figura 33). A Figura 22 mostra a tela de login/cadastro do usuário.

4.2 Esboço das Interfaces Usadas pelo Administrador

Wireframe/mockup/storyboard das interfaces exclusivas do Administrador.

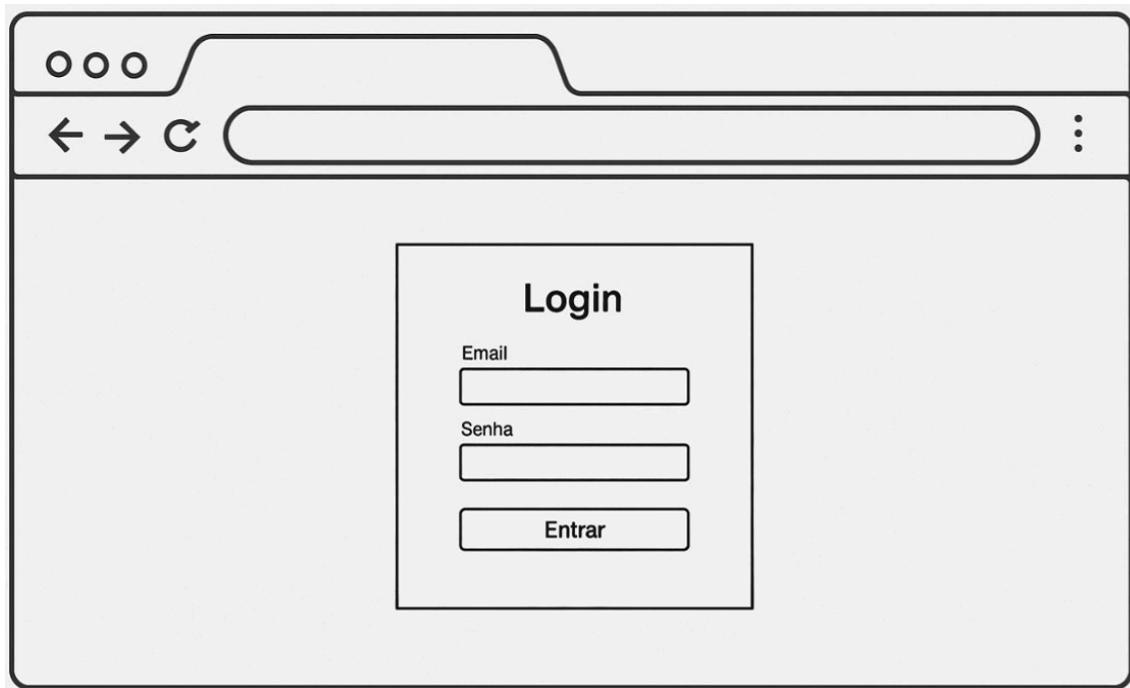


Figura 23. Página de login do administrador.

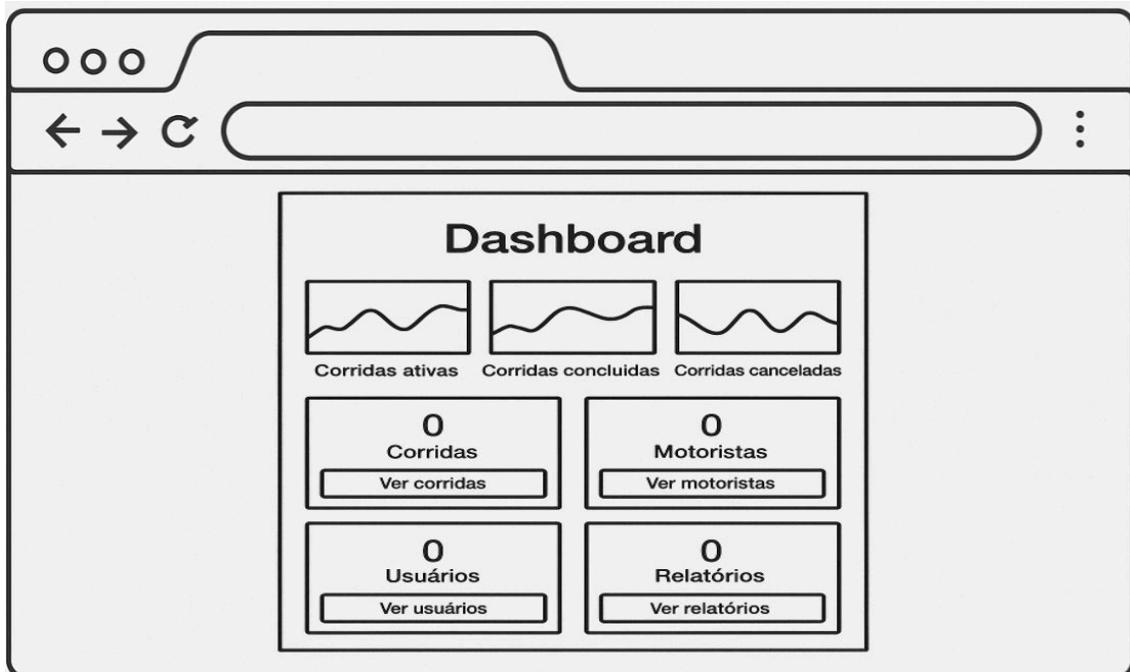


Figura 24. Página do dashboard principal.

A Figura 23 representa a tela de login do administrador, que dá acesso à tela de dashboard principal (Figura 24). O dashboard contém gráficos de corridas ativas, concluídas e canceladas, além de atalhos para funcionalidades administrativas.

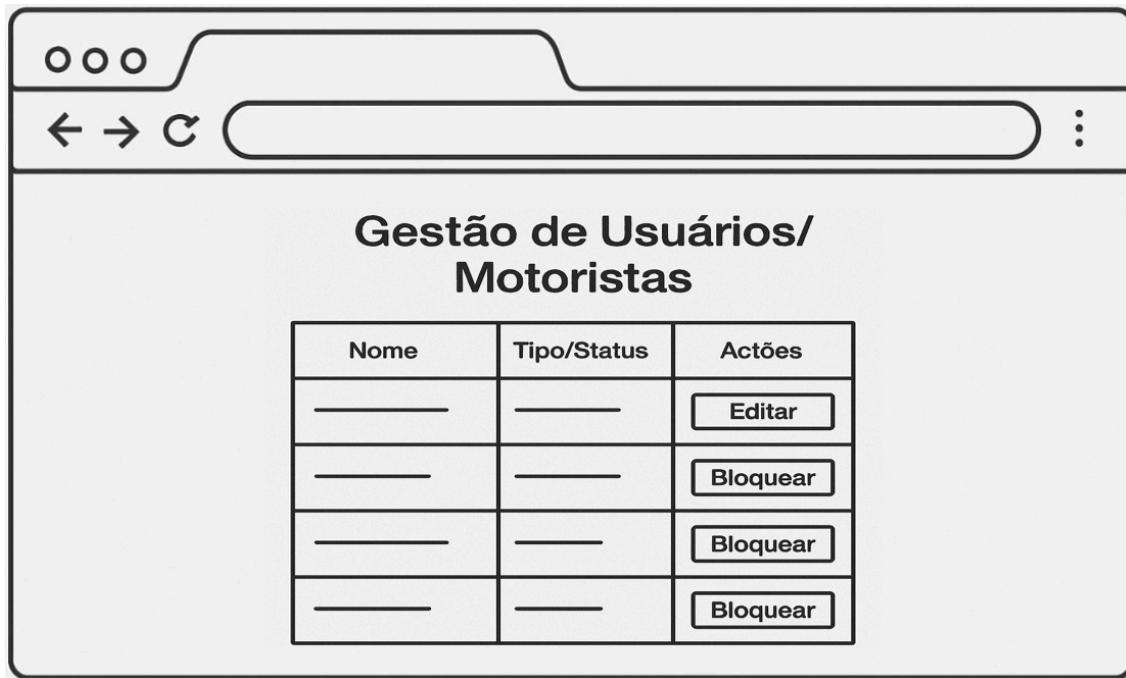


Figura 25. Página da gestão de usuários.

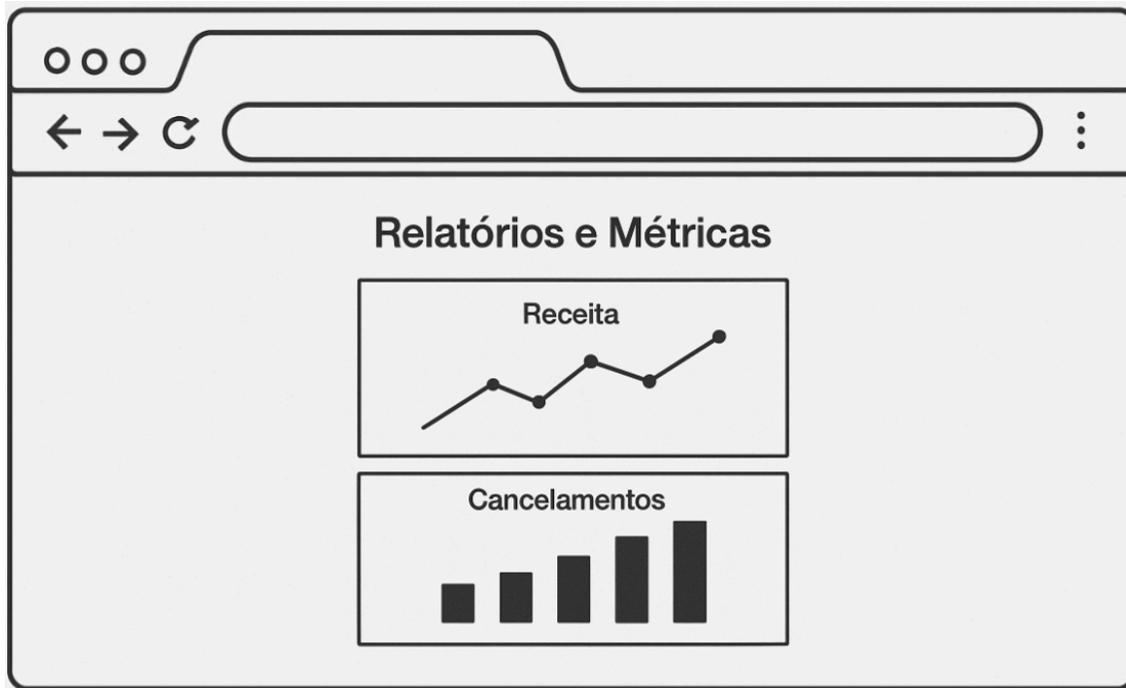


Figura 26. Página de relatórios e métricas.

A Figura 25 apresenta a tela de gestão de usuários e motoristas, com tabela listando os registros, status e botões para editar ou bloquear perfis. A Figura 26 mostra a tela de relatórios e métricas, que contém gráficos sobre receitas, taxas de cancelamento e desempenho do sistema.

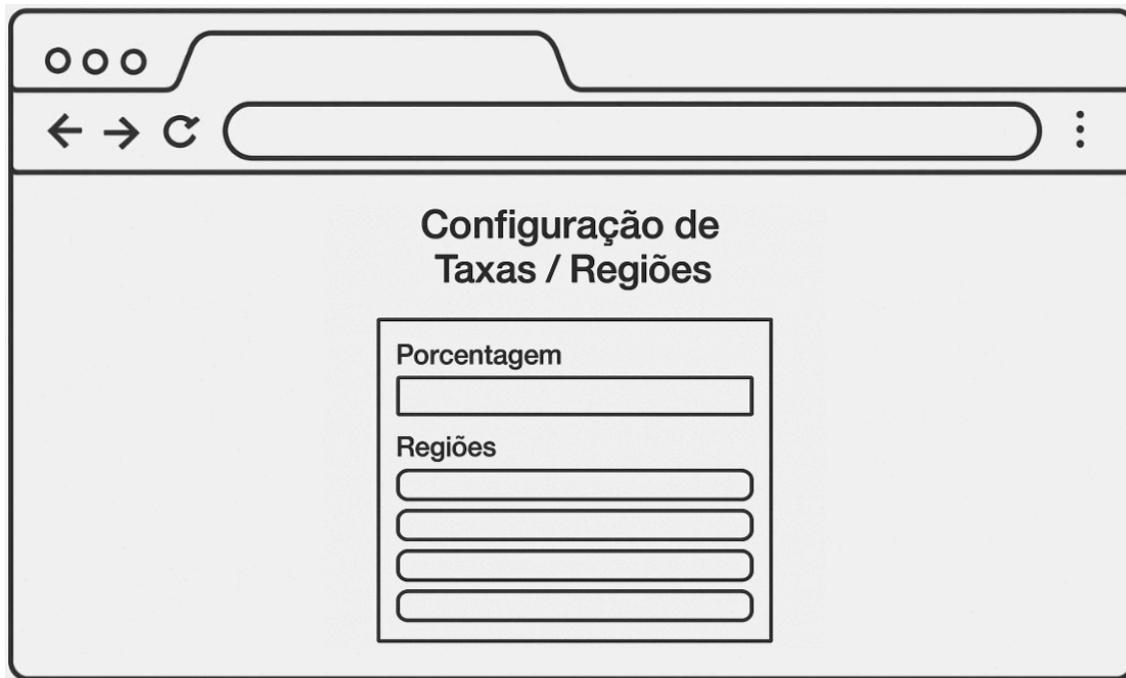


Figura 27. Página de configuração de taxas e regiões.

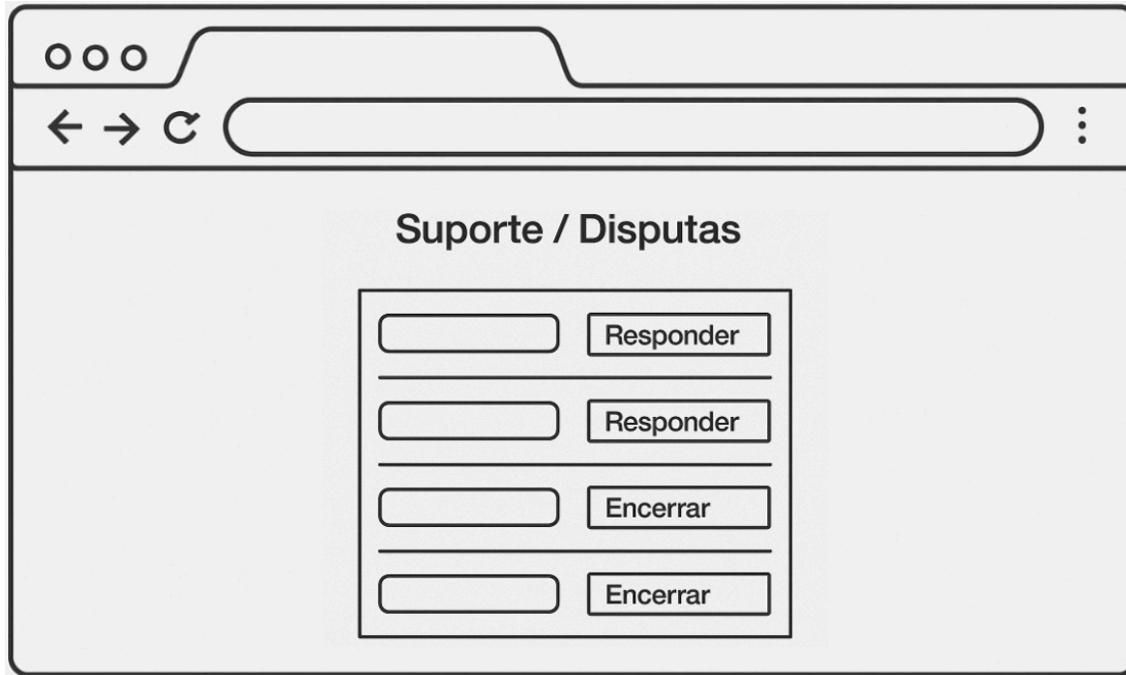


Figura 28. Página de suporte e disputas.

A Figura 27 apresenta a tela de configuração de taxas e regiões, na qual o administrador define percentuais de comissão e gerencia as áreas atendidas pela plataforma. Por fim, a Figura 28 exibe a tela de suporte e disputas, que lista chamados abertos por usuários ou motoristas e permite ao administrador registrar respostas ou encerrar cada caso.

4.3 Esboço das Interfaces Usadas pelo Motorista

Wireframe/mockup/storyboard das interfaces exclusivas do Motorista.



Figura 29. Página de status do motorista.



Figura 30. Página de nova corrida.

Após acessar a conta, o motorista é direcionado para a tela de status online/offline (Figura 29), que possui um botão central destacando o estado atual (verde para disponível, vermelho para indisponível). Quando o motorista está online e recebe uma nova corrida, é exibida a tela de solicitação (Figura 30), que contém os dados do passageiro, origem, destino e valor estimado, além dos botões “Aceitar” e “Recusar”.



Figura 31. Página de navegação do motorista.



Figura 32. Página relatório de ganhos.

Uma vez aceita a corrida, o app abre a tela de navegação (Figura 31), que mostra no mapa a rota até o local de embarque e, posteriormente, até o destino do passageiro. Nesta tela, o motorista também tem acesso aos botões “Cheguei”, “Iniciar viagem” e “Finalizar viagem” (Figura 31), cada um alterando o estado da corrida. Concluída a viagem, o motorista é redirecionado à tela de relatórios de ganhos (Figura 32), que apresenta dados de corridas finalizadas, ganhos do dia e gráficos semanais.

4.4 Esboço das Interfaces Usadas pelo Passageiro

Wireframe/mockup/storyboard das interfaces exclusivas do Passageiro.



Figura 33. Página inicial do passageiro.



Figura 34. Página estimativa de corrida.

A Figura 33 representa a tela principal de solicitação de corrida. Nela o passageiro vê um mapa com a posição atual, campo para inserir o destino e botão para solicitar corrida. Após preencher o destino, o usuário é redirecionado para a tela de estimativa de tarifa (Figura 34), que mostra o valor previsto da viagem e tempo de chegada do motorista. A partir desta tela, o passageiro pode confirmar ou cancelar a solicitação.



Figura 35. Página de acompanhamento.

A Figura 35 apresenta a tela de acompanhamento da corrida, que exibe no mapa a posição do motorista em tempo real, dados do motorista e do veículo, além de um botão para cancelamento da corrida antes do início.

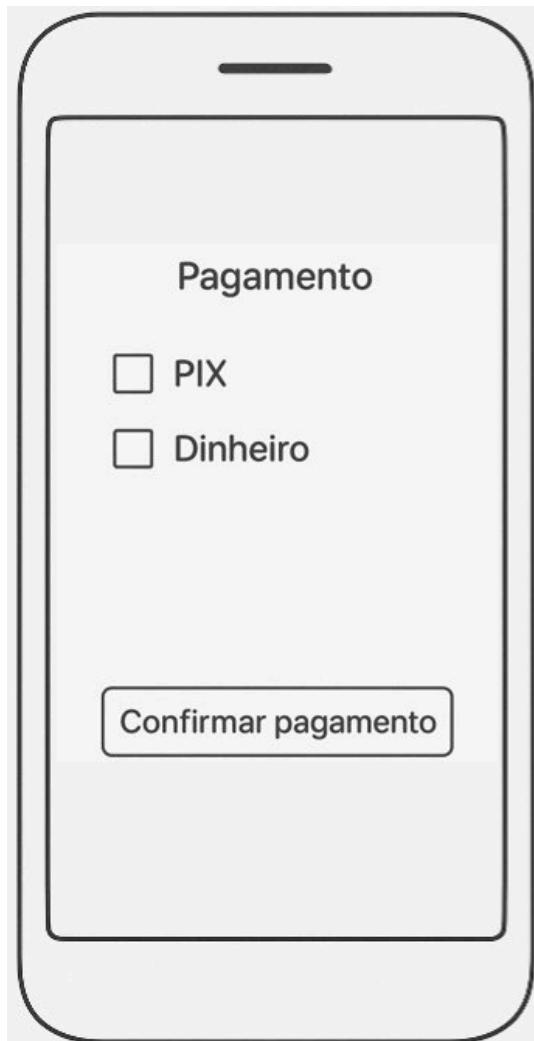


Figura 36. Página forma de pagamento.



Figura 37. Página perfil do usuário.

Após a finalização da viagem, o passageiro é redirecionado à tela de pagamento (Figura 36), onde escolhe entre PIX ou dinheiro. A Figura 37 mostra a tela de perfil do usuário, na qual é possível editar informações pessoais, acessar o histórico de corridas (Figura 37) e encerrar a sessão. O histórico apresenta uma lista de viagens anteriores com informações resumidas como data, valor e status.

5. Glossário e Modelos de Dados

Deve-se apresentar o glossário para o sistema. Também apresente esquemas de banco de dados e as estratégias de mapeamento entre as representações de objetos e não-objetos.

6. Casos de Teste

Uma descrição de casos de teste para validação do sistema.

7. Cronograma e Processo de Implementação

Uma descrição do cronograma para implementação do sistema e do processo que será seguido durante a implementação.