

ASCENDINO MARTINS DE AZEVEDO NETO

JOHNATAN RODRIGUES DOS SANTOS

LARYSSA FINIZOLA COSTA DA SILVA

RELATÓRIO - RELEASE 02

CAMPINA GRANDE – PB

22 de outubro de 2025

1. INTRODUÇÃO

A Release 2 do sistema UberPB representa a segunda etapa de evolução do projeto, consolidando e expandindo as funcionalidades desenvolvidas anteriormente. Nesta fase, o foco principal foi o aperfeiçoamento da experiência de uso e a ampliação dos recursos de pagamento, avaliação e histórico de corridas, de modo a aproximar o sistema de um ambiente mais realista e completo.

Com base na arquitetura estabelecida na Release 1, a equipe deu continuidade à implementação dos requisitos 13 a 18, distribuídos em duas sprints. Entre as principais entregas estão o módulo de pagamentos, com múltiplas formas de transação; o cálculo automatizado do valor da corrida considerando distância, tempo e tarifa dinâmica; a emissão de recibos eletrônicos; e o sistema de avaliações e histórico de corridas, que aprimora a interação e a confiabilidade entre motoristas e passageiros. Assim, esta entrega consolida a maturidade do sistema UberPB, garantindo maior consistência, rastreabilidade e usabilidade.

2. DESCRIÇÃO DA ARQUITETURA

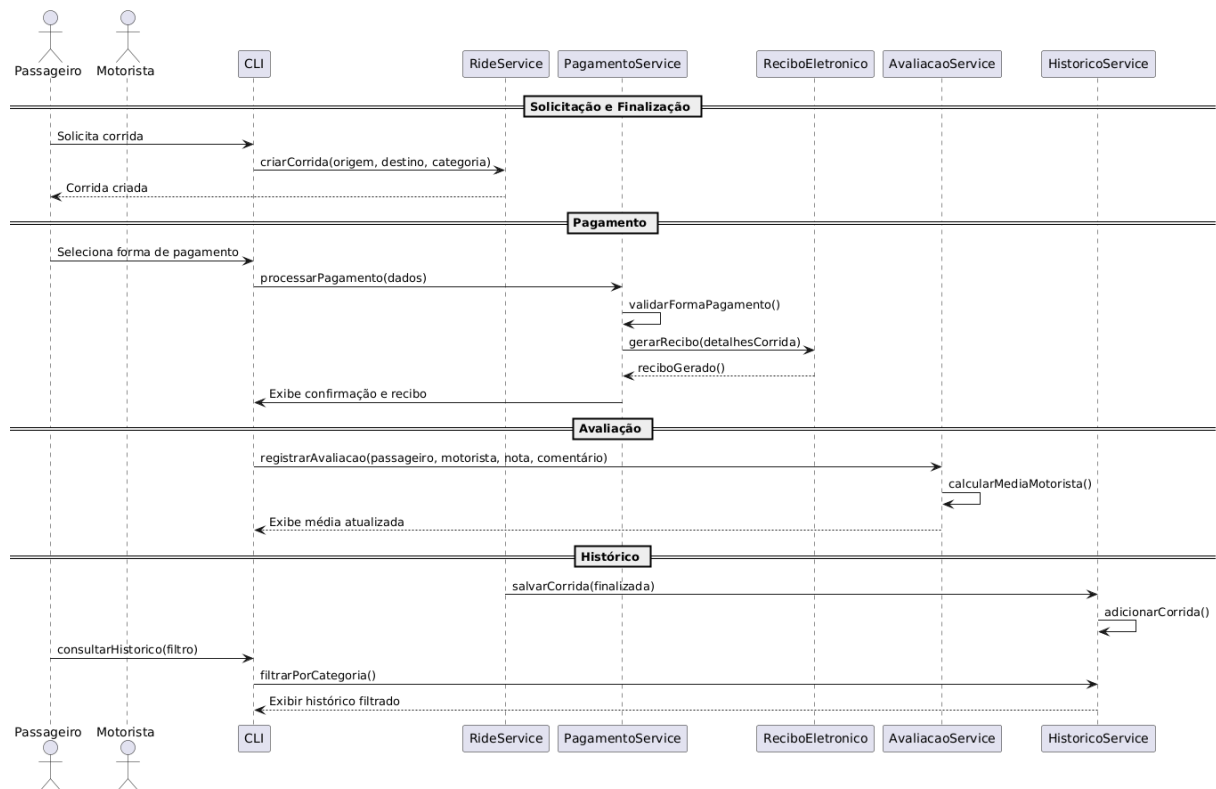
O UberPB mantém sua arquitetura em camadas, adotando uma divisão clara entre modelos, repositórios, serviços e interface. Essa estrutura foi preservada e aprimorada nesta release para suportar os novos módulos e facilitar a integração entre componentes.

Na Release 2, novas classes foram adicionadas à camada de model para representar as entidades relacionadas a pagamento, recibos e avaliações. Entre elas destacam-se as classes Pagamento, Recibo Eletronico, Avaliacao e Historico de Corrida. Na camada de service, surgiram novos serviços responsáveis por coordenar essas entidades:

- **PaymentService:** gerencia os métodos de pagamento via cartão, PIX, PayPal e dinheiro;
- **PricingService:** amplia o cálculo de preço, considerando distância, tempo, categoria do veículo e tarifa dinâmica;
- **RatingService:** administra o registro e a média das avaliações entre passageiros e motoristas;
- **DigitalReceiptService:** gera o recibo de corrida com informações de categoria, forma de pagamento e local.

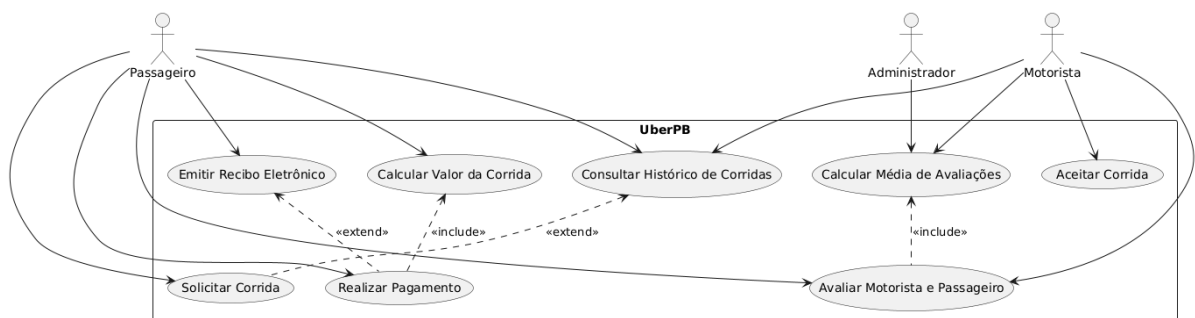
A camada de persistência (repo) continua sendo baseada em arquivos locais, mas agora armazena também informações de pagamentos, avaliações e recibos. Já a camada de interface (cli) foi atualizada para incluir novos fluxos de interação, como a escolha do método de pagamento, emissão de recibo e exibição do histórico de corridas. Essas extensões mantêm o princípio da separação de responsabilidades e reforçam a modularidade, garantindo escalabilidade para futuras releases.

Figura 01: Diagrama de Sequência.



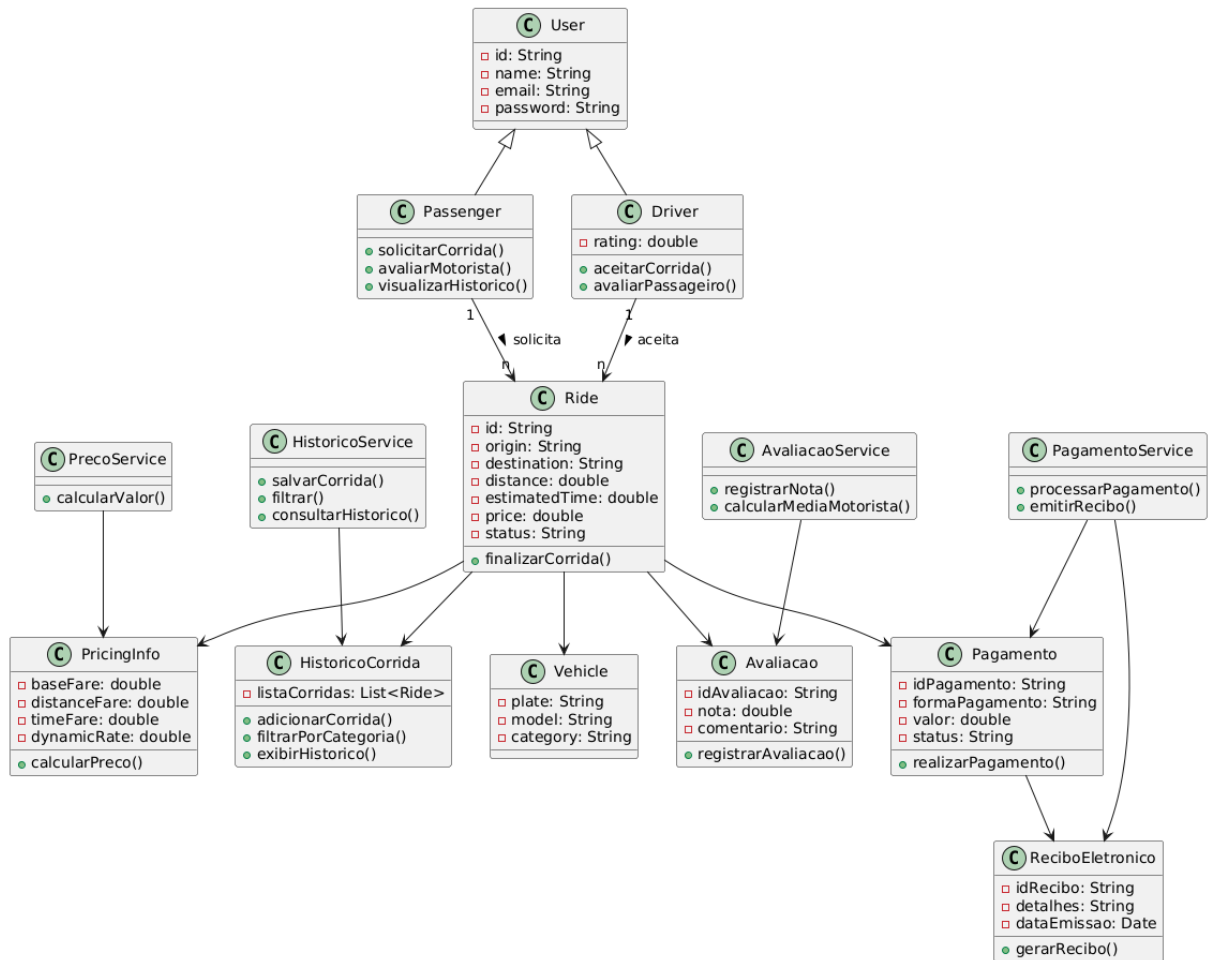
Fonte: Autoria Própria, 2025

Figura 02: Diagrama de Caso de Uso.



Fonte: Autoria Própria, 2025.

Figura 03: Diagrama de Classes.



Fonte: Autoria Própria, 2025.

3. FUNCIONALIDADES DESENVOLVIDAS

3.1 Pagamentos e Recibos Eletrônicos

O módulo de pagamento foi uma das principais adições desta release, oferecendo quatro opções de pagamento: cartão, PIX, PayPal e dinheiro. O processo é mediado pelo **PaymentService**, que simula a integração com APIs de transação, validando os dados e confirmando o recebimento. Cada transação gera um recibo eletrônico por meio da classe **Receipt**, contendo informações essenciais como identificação da corrida, motorista, passageiro, valor total e forma de pagamento. Essa funcionalidade aumenta a transparência e profissionalismo do sistema. O recibo é armazenado localmente e pode ser consultado posteriormente, garantindo rastreabilidade e controle financeiro para ambas as partes.

O cálculo de preço foi aprimorado com base em quatro parâmetros principais: distância percorrida, tempo estimado, categoria do veículo e tarifa dinâmica. O PricingService centraliza esse processo, aplicando fórmulas que variam conforme a categoria selecionada (básico, conforto, premium) e o horário da solicitação, que pode influenciar a tarifa dinâmica. Esse mecanismo permite simular flutuações reais de preço, refletindo condições típicas de mercado, como demanda e distância. A integração com as classes Tariff e VehicleCategory garante que os resultados sejam consistentes com as políticas do sistema.

Figura 04: Solicitando a corrida mostrando o preço dinâmico.

```

=== Solicitar Corrida (RF04 + RF05) ===
Email do passageiro: teste@teste.com
Endereço de origem: Rua Samuel Araujo Diniz, 183
Endereço de destino: UEPB - Campus I

=== Estimativas de Corrida ===
Origem: Rua Samuel Araujo Diniz, 183
Destino: UEPB - Campus I
Distância estimada: 3,4 km
Tempo estimado: 14 min

1. UberX - R$ 10,78 (14 min)
2. Uber Bag - R$ 11,70 (14 min)
3. Uber Comfort - R$ 12,65 (13 min)
4. Uber XL - R$ 14,82 (13 min)
5. Uber Black - R$ 16,80 (12 min)
Escolha uma opção (1-5): 3

--- Formas de Pagamento (RF13) ---
1 - Cartão de Crédito
2 - PIX
3 - PayPal
4 - Dinheiro
Escolha uma forma de pagamento: 2
Nenhum motorista disponível. A corrida ficará solicitada.

=== Corrida Solicitada com Sucesso! ===
ID da corrida: a04cdd4f-2779-4d33-ac57-ff9bafef1a6e8
Categoria escolhida: Uber Comfort
Preço estimado: R$ 12,65
Tempo estimado: 13 min
Forma de Pagamento: PIX
Status: Solicitada

```

Figura 05: Pagamento de Corrida.

```


=== Pagar Corrida (RF13) ===
Digite o ID da corrida: 73273603-e581-4df8-b7b1-ae4ceca51ff6

=====
PROCESSAMENTO DE PAGAMENTO
=====
Valor: R$ 48,30
Método: PIX
=====

PAGAMENTO PIX

Chave PIX: 3f631320... |

QR CODE PARA PAGAMENTO:



INSTRUÇÕES:
1. Abra seu app de pagamentos
2. Escaneie o QR Code acima
3. Ou digite a chave PIX: 3f631320-cf15-430d-bd67-f7890c1e7b6e
4. Confirme o pagamento

Após realizar o pagamento, pressione ENTER para confirmar...
Pagamento confirmado!
  
```

Fonte: Eclipse IDE.

Figura 06: Finalizando e gerando Recibo.

```

=== Gerar / Enviar / Visualizar Recibo (RF15) ===
Digite o ID da corrida: a04cdd4f-2779-4d33-ac57-ff9bafela6e8
Forma de pagamento (ou deixe em branco para 'Não informado'): PIX
Recibo gerado e salvo em: receipts\04cdd4f-2779-4d33-ac57-ff9bafela6e8-receipt-a04cdd4f-2779-4d33-ac57-ff9bafela6e8-336eef70.txt
Recibo enviado para: teste@teste.com (simulação).
Recibos encontrados:
1) a04cdd4f-2779-4d33-ac57-ff9bafela6e8-receipt-a04cdd4f-2779-4d33-ac57-ff9bafela6e8-336eef70.txt
Escolha número para visualizar (ou ENTER para voltar): 1

--- Conteúdo do Recibo ---

=== RECIBO ELETRÔNICO ===
Recibo ID: receipt-a04cdd4f-2779-4d33-ac57-ff9bafela6e8-336eef70
Corrida ID: a04cdd4f-2779-4d33-ac57-ff9bafela6e8
Gerado em: 2025-10-24T01:14:25.390404700Z

--- Passageiro ---
Nome: Teste
Email: teste@teste.com

--- Motorista ---
Nome: testemotorista@teste.com
Email: testemotorista@teste.com

--- Corrida ---
Origem: Rua Samuel Araujo Diniz, 183
Destino: UEPB - Campus I
Distância estimada: 3,4 km
Tempo estimado: 13 min
Categoria: Uber Comfort
Status da corrida: Finalizada

--- Pagamento ---
Forma: PIX
Valor total: R$ 12,65

Obrigado por utilizar o serviço.
  
```

Fonte: Eclipse IDE.

3.2 Sistema de Avaliações

Para fortalecer a confiabilidade e a qualidade do serviço, foi criado o sistema de avaliações mútuas entre passageiros e motoristas. As médias são calculadas automaticamente pelo RatingService e persistidas para uso futuro, de modo que motoristas com melhores avaliações sejam priorizados em corridas de categorias superiores. Essa abordagem introduz um elemento de meritocracia e incentiva a boa conduta dos usuários, tanto motoristas quanto passageiros.

Figura 07: Modo de avaliar a corrida.

```
=== Avaliar Corrida (RF16) ===
Digite o ID da corrida que deseja avaliar: a04cdd4f-2779-4d33-ac57-ff9bafef1a6e8
Digite seu email para identificação: teste@teste.com
Qual sua nota (de 1 a 5)? 5
Motorista avaliado com sucesso!
```

Fonte: Eclipse IDE.

3.3 Histórico de Corridas

O histórico de corridas foi ampliado para incluir filtros por categoria de veículo e status. A classe RideHistory armazena informações completas sobre cada viagem, permitindo que o usuário consulte facilmente corridas anteriores. A interface de linha de comando foi atualizada para permitir filtragem dinâmica, exibindo apenas as corridas que atendam aos critérios selecionados (por exemplo, apenas “premium” ou “em andamento”). Essa funcionalidade melhora a experiência de uso e a organização dos dados de cada perfil.

Figura 08: Histórico de corridas.

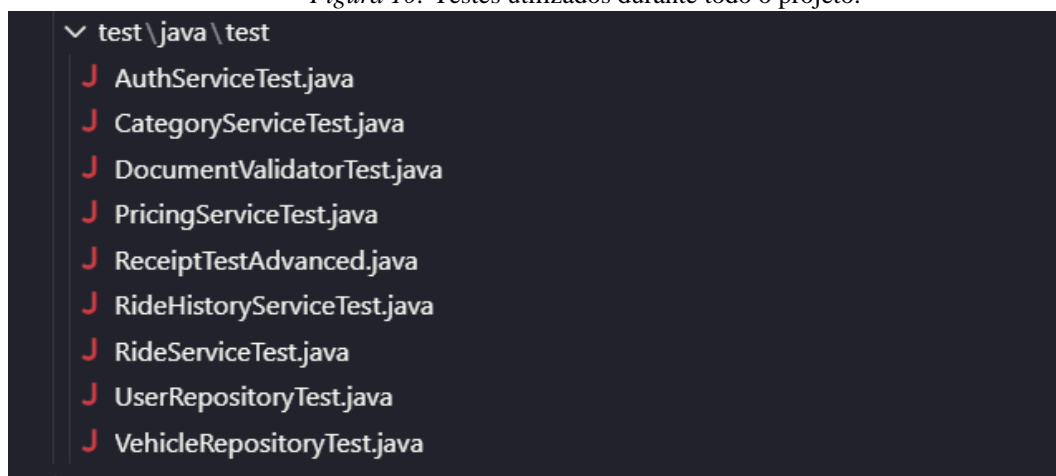
```
=== HISTÓRICO DE CORRIDAS ===
Total de registros: 3
1. Histórico[ID=5087663f-f3e6-4321-af72-60b1a29fd9ae, Passageiro=teste@teste.com, Motorista=testemotorista@teste.com, Origem=Rua Samuel Araujo Diniz, 183, De
2. Histórico[ID=5419be7b-3887-4c04-96e1-26a64e214cf9, Passageiro=teste@teste.com, Motorista=testemotorista@teste.com, Origem=rua teste 1, Destino=rua de test
3. Histórico[ID=18ea4219-86aa-4ce4-8988-ed2c13e658d1, Passageiro=teste@teste.com, Motorista=testemotorista@teste.com, Origem=rua teste 1, Destino=rua de test
=====
```

Fonte: Eclipse IDE.

3. TESTES E COBERTURA DE CÓDIGO

A Release 2 manteve o compromisso com a qualidade e confiabilidade do código, garantindo uma cobertura mínima de 80% em testes unitários com o framework JUnit, utilizando o Emma Plugin para geração dos relatórios. Entre os testes implementados, destacam-se o `ReceiptTestAdvanced`, que confirma a geração e os dados contidos nos recibos, e o `RideServiceTest`, que assegura a persistência e filtragem correta do histórico de corridas. Esses testes cobrem os fluxos críticos da aplicação, evitando regressões e assegurando estabilidade mesmo com o aumento da complexidade do sistema.

Figura 10: Testes utilizados durante todo o projeto.



Fonte: Eclipse IDE.

4. CONCLUSÃO

Durante o desenvolvimento da Release 2, foram implementados os requisitos 13 a 18, consolidando o UberPB como um sistema funcional e robusto. A equipe, composta por Ascendino Martins de Azevedo Neto, Johnatan Rodrigues dos Santos e Laryssa Finizola Costa da Silva, trabalhou em duas sprints (Sprint 3: 25/09 – 08/10 e Sprint 4: 09/10 – 22/10), sob a gerência alternada de Laryssa e Johnatan, garantindo a execução organizada das tarefas e o cumprimento dos prazos. A entrega desta release ampliou significativamente as capacidades do sistema, incorporando mecanismos de pagamento, recibos, avaliações e histórico detalhado. Além disso, manteve-se o uso da linguagem Java, com arquitetura em camadas, padrões de projeto (Design Patterns) e persistência local em arquivos.

5. Referências

PLANTTEXT. PlantText: PlantUML Editor Online – Free & Fast UML Diagram Tool. Disponível em: <https://www.planttext.com/> Acesso em: 24 out. 2025.

WINTERS, Titus; MANSHRECK, Tom; WRIGHT, Hyrum. Software engineering at Google: lessons learned from programming over time. 1. ed. Sebastopol: O'Reilly Media, 2020.