# Relatório: UberPB

# O backlog com os itens de trabalho e as tasks;

Para apresentação do backlog e as task será utilizado prints do Azure DevOps

* **Sprint 1 (26/08 – 08/09)**

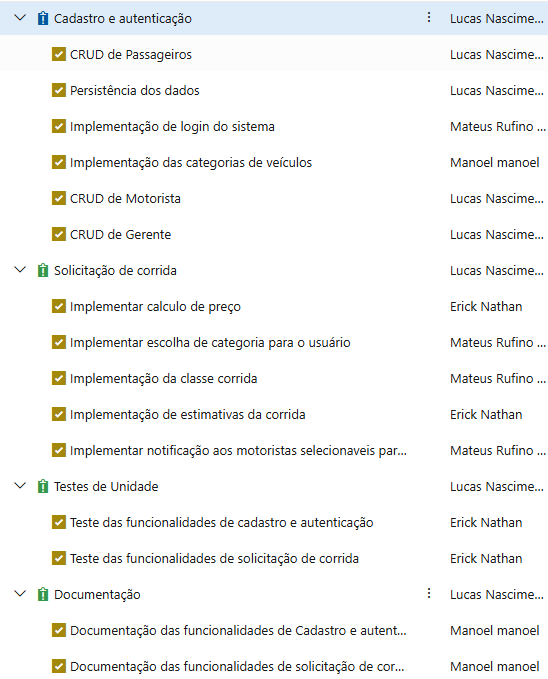
****

Figura 1

Durante a Sprint 1, a equipe concentrou-se no desenvolvimento inicial das principais funcionalidades do sistema, garantindo a base para as próximas entregas. As atividades realizadas foram:

* + **Cadastro e autenticação:** CRUD de Passageiros, Persistência dos dados, Implementação de login do sistema, Implementação das categorias de veículos, CRUD de Motorista e CRUD de Gerente.
  + **Solicitação de corrida:** Cálculo de preço, Escolha de categoria para o usuário, Classe corrida, Estimativas da corrida e Notificação aos motoristas selecionados.
  + **Testes de unidade:** Testes das funcionalidades de cadastro/autenticação e solicitação de corrida.
  + **Documentação:** Registro das funcionalidades de Cadastro e Autenticação, além da Solicitação de Corrida.

Todos os itens planejados foram concluídos com sucesso, permitindo validar as primeiras interações entre usuários e o sistema.

* + **Papéis**
    - **Erick:** Testador
    - **Lucas:** Gerente, Desenvolvedor
    - **Manoel:** Desenvolvedor, Engenheiro de software
    - **Mateus:** Desenvolvedor

Como foi implementado (Cadastro e Autenticação): Foram criadas as classes de entidade (Passageiro, Motorista, Gerente), os Repositórios (PassageiroRepository, MotoristaRepository) e os Serviços (PassageiroService, MotoristaService) para lidar com as regras de negócio e a persistência em arquivos JSON. A enum CategoriaVeiculo foi criada e o LoginView foi implementado para autenticação.

Como foi implementado (Solitação de corrida): A Corrida.java foi criada. A lógica de cálculo de preço foi adicionada ao CorridaService. A SolicitarCorridaView foi criada para permitir ao usuário inserir origem/destino e escolher a categoria. A notificação foi implementada no método encontrarProximoMotoristaDisponivel do CorridaService.

* **Sprint 2 (09/09 – 22/09)**

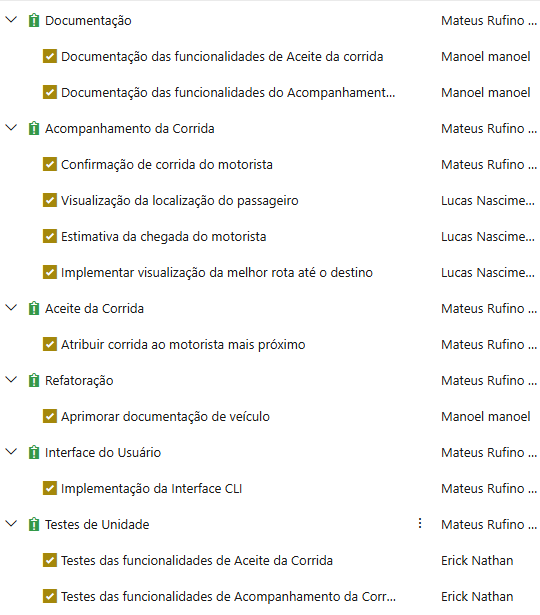
****

Figura 2

Na Sprint 2, o foco foi ampliar as funcionalidades já desenvolvidas, adicionando camadas de acompanhamento e aceitação da corrida, além de melhorias na interface e na documentação. As atividades realizadas foram:

* + **Documentação:** Funcionalidades de Aceite da Corrida e Acompanhamento da Corrida.
  + **Acompanhamento da Corrida:** Estimativa da chegada do motorista e visualização da melhor rota até o destino.
  + **Aceite da Corrida:** Confirmação da corrida pelo motorista e atribuição da corrida ao motorista mais próximo.
  + **Refatoração:** Melhoria da documentação de veículos.
  + **Interface do Usuário:** Implementação da Interface CLI.
  + **Testes de unidade:** Testes das funcionalidades de Aceite da Corrida e Acompanhamento da Corrida.

Grande parte das tarefas foi concluída, enquanto as atividades com os requisitos com funcionalidade de localização do motorista ainda faltam terminar a implementação.

* + **Papéis**
    - **Erick:** Testador
    - **Lucas:** Desenvolvedor
    - **Manoel:** Desenvolvedor, Engenheiro de software
    - **Mateus:** Gerente, Desenvolvedor

Como foi implementado (Aceite corrida): A lógica de atribuição foi implementada no CorridaService, que filtra motoristas por status e categoria, e os ordena por distância/avaliação. A confirmação foi implementada nos métodos aceitarCorrida e negarCorrida, acessíveis ao motorista pela CorridaView.

Como foi implementado (Acompanhamento da corrida): A visualização da rota foi implementada através do SimuladorViagem.java, que atualiza um dados\_viagem.js e é exibido no Mapa.html. A AcompanharCorridasView foi criada para o passageiro ver o status.

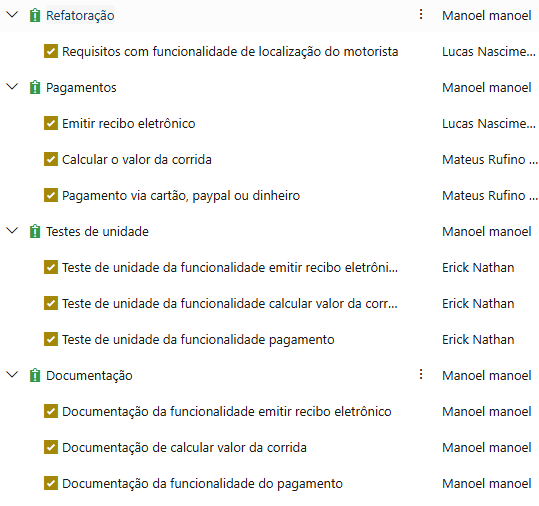
* **Sprint 3 (25/09 – 08/10)**

Figura 3

Durante a Sprint 3, o foco da equipe esteve na implementação das funcionalidades relacionadas a **pagamentos**, **refatoração de código**, **testes de unidade** e **documentação**. Todas as atividades planejadas foram concluídas dentro do prazo estabelecido, garantindo a continuidade do desenvolvimento com qualidade e estabilidade.  
  
As principais entregas desta sprint foram:

* + **Refatoração**: Ajustes e melhorias no código, incluindo a adição de requisitos com a funcionalidade de localização do motorista.
  + **Pagamentos**: Implementação das funcionalidades de emissão de recibo eletrônico, cálculo do valor da corrida e integração com múltiplas formas de pagamento (cartão, PayPal e dinheiro).
  + **Testes de unidade**: Desenvolvimento e execução de testes para validar as funcionalidades de emissão de recibo eletrônico, cálculo de valor da corrida e pagamento.
  + **Documentação**: Registro técnico das funcionalidades implementadas, abrangendo pagamento, cálculo de valor da corrida e emissão de recibo eletrônico.

A sprint foi finalizada com 100% das atividades concluídas, consolidando uma base sólida para a próxima etapa do projeto, com foco em garantir qualidade e manutenção das novas funcionalidades.

* + **Papéis**
    - **Erick:** Testador
    - **Lucas:** Desenvolvedor
    - **Manoel:** Gerente, Engenheiro de software
    - **Mateus:** Desenvolvedor

Como foi implementado (Refatoração): A lógica de simulação de localização do motorista foi concluída no MotoristaService e CorridaView, permitindo ao motorista "mudar" de local e receber corridas com base nessa nova posição.

Como foi implementado: A PagamentoView.java foi criada para processar as formas de pagamento . Além das formas citadas na *task*, Pix também foi implementado. O Cálculo do valor foi refinado no CorridaService. O recibo eletrônico é apresentado ao usuário no HistoricoCorridasView.

* **Sprint 4 (9/10 – 22/10)**

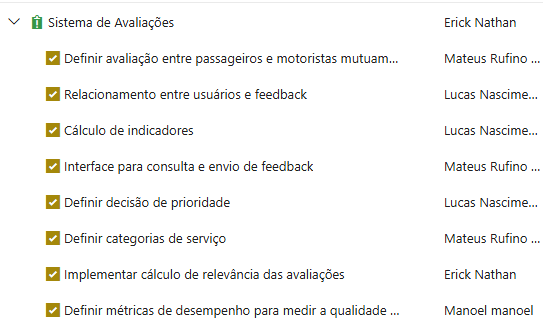


Figura 4

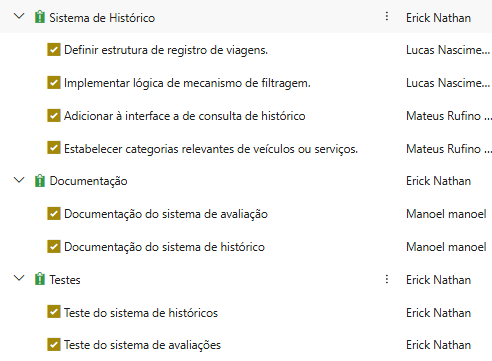


Figura 5

Durante a Sprint 4, a equipe concentrou seus esforços no desenvolvimento e documentação das novas funcionalidades de **avaliações** e **histórico de corridas**, ampliando a experiência dos usuários e aprimorando a interação entre passageiros e motoristas. Todas as atividades planejadas foram concluídas com sucesso, garantindo a entrega completa dos requisitos definidos.  
As principais entregas desta sprint foram:

* + **Avaliação**: Foi desenvolvida a funcionalidade que permite que passageiros e motoristas realizem avaliações mútuas após o término de cada corrida. Além disso, as médias de avaliação passaram a ser consideradas no momento de priorização dos motoristas.
  + **Histórico**: Implementação do histórico de corridas, que possibilita o registro completo das viagens realizadas pelos usuários. A funcionalidade permite filtrar corridas por categoria de veículo, facilitando a consulta e o acompanhamento de viagens anteriores.
  + **Documentação**: Registro técnico das funcionalidades implementadas, abrangendo avaliação mútua, priorização dos motoristas por avaliação e o histórico das corridas.

A sprint foi finalizada com todas as atividades concluída**s**, garantindo estabilidade e ampliando as funcionalidades do sistema com foco em usabilidade e qualidade.

* + **Papéis**
    - **Erick:** Gerente, Testador
    - **Lucas:** Desenvolvedor
    - **Manoel:** Engenheiro de software
    - **Mateus:** Desenvolvedor

Como foi implementado (Avaliação): A avaliação mútua foi implementada nas classes AvaliacaoView e AvaliacaoPendentesView. A priorização de motoristas foi implementada no CorridaService, que usa um comparador para ordenar motoristas com base na média de avaliação. A média é recalculada e salva pelos métodos recalcularMediaAvaliacao nos serviços MotoristaService e PassageiroService.

Como foi implementado (Histórico): A Interface de consulta e a lógica de filtragem por categoria foram implementadas na classe HistoricoCorridasView. A estrutura de registro é a lista historicoCorridas na entidade Passageiro.java.