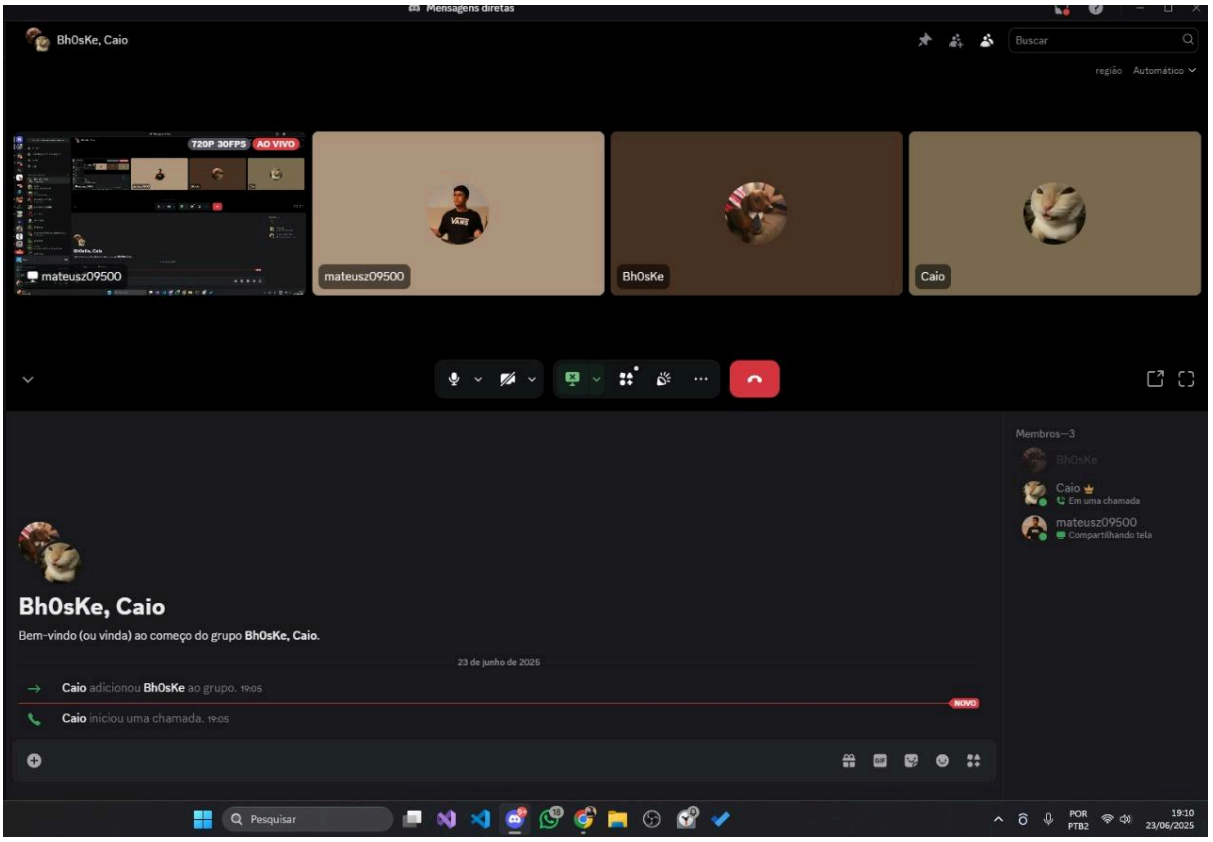
Sistema de Logística de Entrega de Mercadorias

Integrantes: *Caio Kfuri, Mateus Ribas e Bruno Hoske*

O objetivo do sistema de Logística de Entrega de Mercadorias (SLEM) é simular o gerenciamento de pedidos, veículos e locais, com foco na manipulação eficiente dos dados e na implementação de um algoritmo básico para cálculo de rotas de entrega. Adotamos também a metodologia **SCRUM** para organizar o desenvolvimento em **sprints semanais**, com tarefas distribuídas entre os membros da equipe. Realizamos uma primeira reunião remota para definir o escopo do protótipo e, nas duas sprints seguintes, realizamos encontros presenciais com o orientador para tirar dúvidas e com o feedback recebido, continuar com a codificação do projeto.

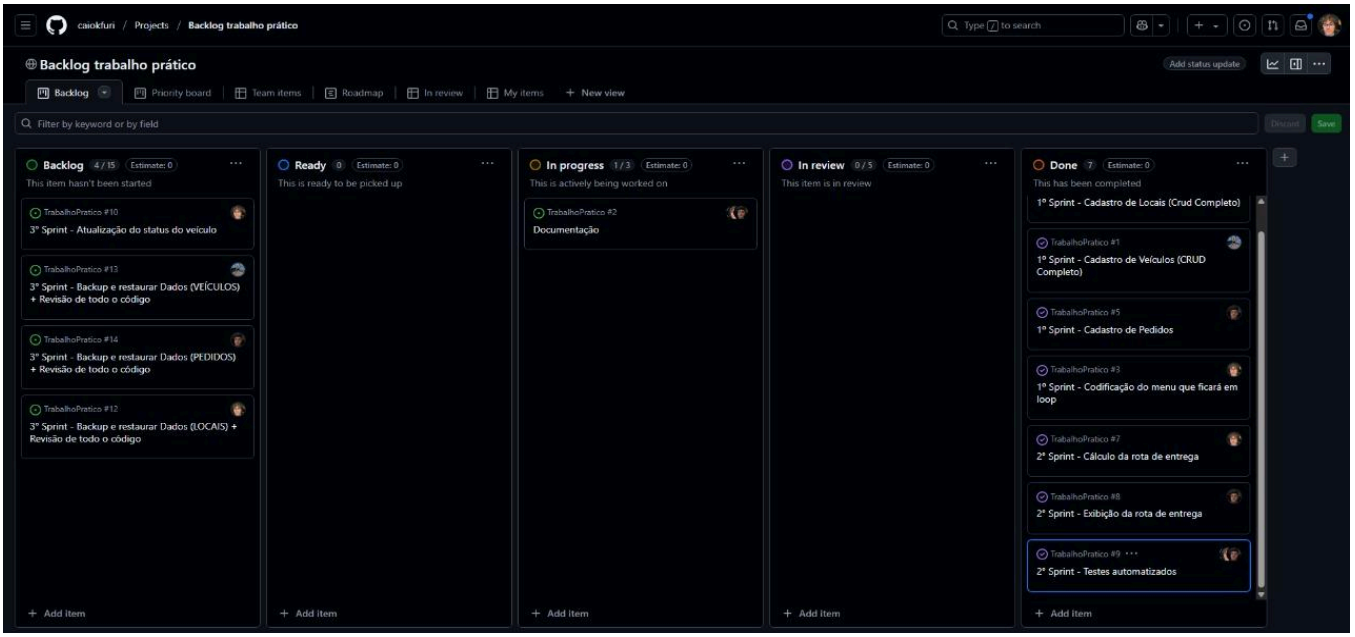
Reunião remota -



✅ SPRINT 1

A Sprint 1 teve como objetivo principal a implementação dos cadastros básicos (**CRUDS**) e da estrutura inicial do sistema. Nesta etapa, cada integrante ficou responsável pelo CRUD de cada classe (Locais, veículos e pedidos), e também foi implementado a codificação do **menu inicial** com laço de repetição. Conforme mostrado na Figura 1 (Abaixo), todas essas tarefas foram concluídas até o final da Sprint e movidas para a coluna de "Done".

FIGURA 1 -

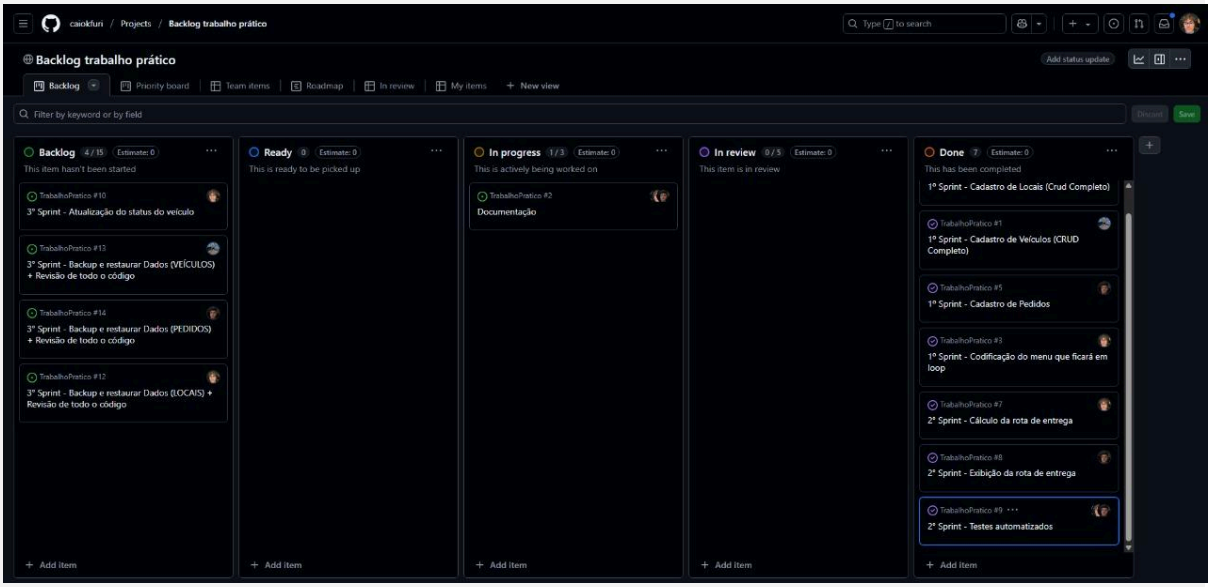


✅ SPRINT 2

Durante a sprint 2, o foco foi voltado para a implementação de funcionalidades relacionadas ao **cálculo e exibição da rota de entrega**, além da criação de **testes automatizados** para garantir a confiabilidade e funcionalidade do sistema. Como visto na figura 2 (abaixo), as atividades foram devidamente organizadas, com progresso visual monitorado em tempo real. A equipe manteve o ritmo e finalizou todas as

tarefas planejadas para essa sprint, consolidando mais uma etapa essencial do projeto.

Figura 2 -



✅ SPRINT 3

A Sprint 3 foi dedicada ao refinamento e manutenção do sistema, e também na implementação de funções para salvar e restaurar dados em arquivos binários. A documentação também entrou como uma tarefa, para ser revisada e completada. As figuras 3 e 4 (ambas abaixo) mostram como o quadro estava com as tarefas no “in progress” e “In review” antes do final da sprint, após o fim dela, todos foram movidos para “Done”.

Figura 3 -

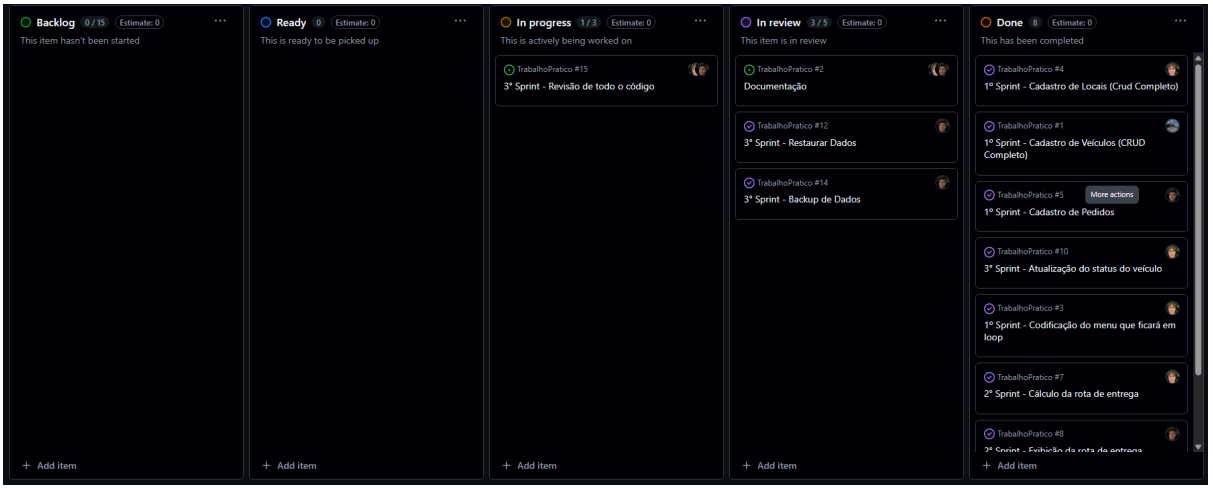


Figura 4 -

(FOTO TUDO NO DONE)

Funções e seus Parâmetros

LocalModel:

preencher() - Chamado para realizar os passos no console e preencher o objeto com os dados necessários.
mostrar() - Chamado para exibir os dados atual do objeto

LocalService:

AdicionarLocal(vector<Local>& locais) - Cria e adiciona um local ao vetor de dados atual

RemoverLocal(vector<Local>& locais) - Remove um local ao vetor de dados atual. (Exibe todos os atuais e solicita que o usuário escolha o índice)

AtualizarLocal(vector<Local>& locais) - Atualiza um Local específico (Exibe todos os atuais e solicita que o usuário escolha o índice)

ListarLocais(vector<Local>& locais) - Lista todos os locais atuais no vetor.

VeiculoModel:

preencher() - Chamado para realizar os passos no console e preencher o objeto com os dados necessários.
mostrar() - Chamado para exibir os dados atual do objeto

VeiculoService:

AdicionarVeiculo(vector<Veiculo>& veiculos) - Cria e adiciona um veículo ao vetor de dados atual

RemoverVeiculo(vector<Veiculo>& veiculos) - Remove um veículo ao vetor de dados atual. (Exibe todos os atuais e solicita que o usuário escolha o índice)

AtualizarVeiculo(vector<Veiculo>& veiculos) - Atualiza um veículo específico (Exibe todos os atuais e solicita que o usuário escolha o índice)

ListarVeiculos(vector<Veiculo>& veiculos) - Lista todos os veículos atuais

---

PedidoModel:

preencher() - Chamado para realizar os passos no console e preencher o objeto com os dados necessários.
mostrar() - Chamado para exibir os dados atual do objeto

PedidoService:

AdicionarPedido(vector<Pedido>& pedidos) - Cria e adiciona um Pedido ao vetor de dados atual

RemoverPedidol(vector<Pedido>& pedidos) - Remove um pedido do vetor de dados atual. (Exibe todos os atuais e solicita que o usuário escolha o índice)

AtualizarPedido(vector<Pedido>& pedidos) - Atualiza um pedido específico (Exibe todos os atuais e solicita que o usuário escolha o índice)

ListarPedidos(vector<Pedido>& pedidos) - Lista todos os pedidos atuais do vetor de dados.

Dados:

BackupDados(const vector<Local>& locais, const vector<Veiculo>& veiculos, const vector<Pedido>& pedidos) - Realiza o backup de dados de todos os dados em arquivos diferentes.

RestaurarDados(const vector<Local>& locais, const vector<Veiculo>& veiculos, const vector<Pedido>& pedidos) - Restaura todos os dados de acordo com o arquivo escolhido

Testes

Cadastro de Veículos:

Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
Nome Veiculo	MeuCarro	Veículo cadastrado com sucesso	“ ”	Nome inválido
Placa	LSH8362	Veículo cadastrado	"LS123"	Placa inválida
Local Atual	Região Metropolitana	Local Do carro	" "	Local não encontrado
Status (Disponível)	0	Veículo Disponível	2	Status inválido

Cadastro de Locais:

Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
Nome do Local	Contagem	Local Adicionado	" "	Nome Vazio
X	28,9	Local Cadastrado	-83,7	Coordenada inválida
Y	49,92	Local Cadastrado	CGWEW EGF	Coordenada inválida

Cadastro de Pedidos -

Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
ID do Pedido	92	Criado com sucesso	-92	ID invalido (Negativo)
Origem	Sudeste	Lugar Correto	“ “	Origem vazia: Erro
Destino	Shopping Mall	Lugar Correto	“ “	Origem vazia: Erro
Peso				

Serviço de Entrega -

Variáveis de Entrada	Valores Válidos	Resultado Esperado	Valores Inválidos	Resultado Esperado
Pedido	id = 382	Lista Veiculos	“ “	Pedido vazio
Frota	status = 0	Veículo Proximo	“ “	Frota Vazia
Locais	Centro	Local Existente	“ “	Local Vazio
Resposta	'S' ou 's'	status atualizado	Outra Letra	Letra Inválida
Pedido	Local/coordenada	Distância	“ “	Distancia Vazia