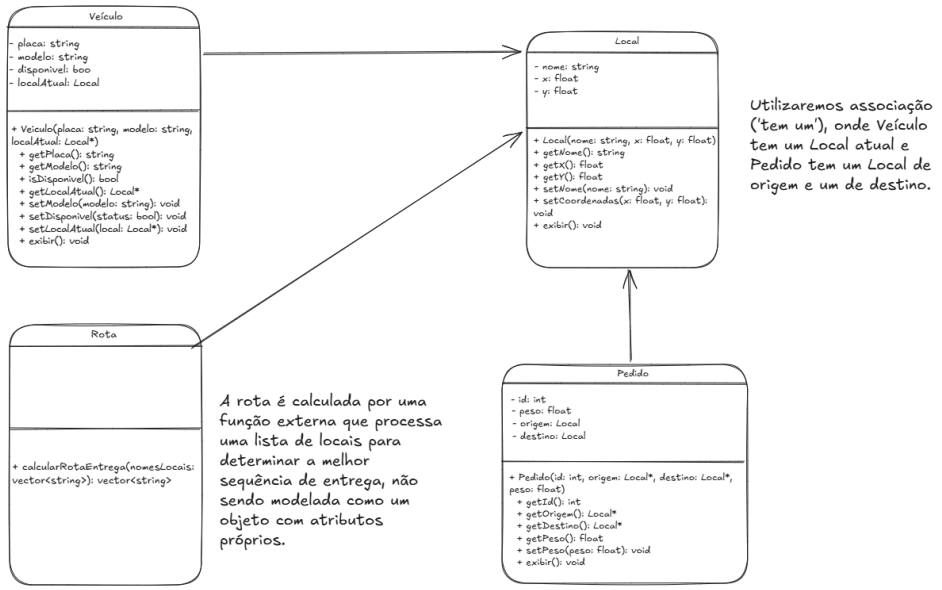
PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS
Graduação em Engenharia de Software

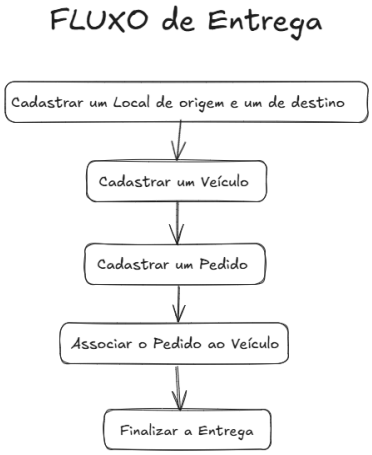
Nome dos integrantes do grupo: Bernardo Leão Braga e Murilo Freitas de Souza
Nome do sistema : Sistema de Logística de Entrega de Mercadorias

Apresentação:
O sistema gerencia uma logística de entregas, permitindo o cadastro e controle de locais, veículos e pedidos. Ele simula o fluxo de uma entrega, desde o cadastro dos dados até a associação de pedidos a veículos, cálculo de rotas e atualização do status dos veículos e pedidos.

Planejamento do Produto
A imagem a seguir mostra um diagrama de planejamento das classes veículo, rota, local e pedido e como elas irão interagir entre si.



Já a imagem a seguir, é uma representação do fluxo de entrega do sistema:

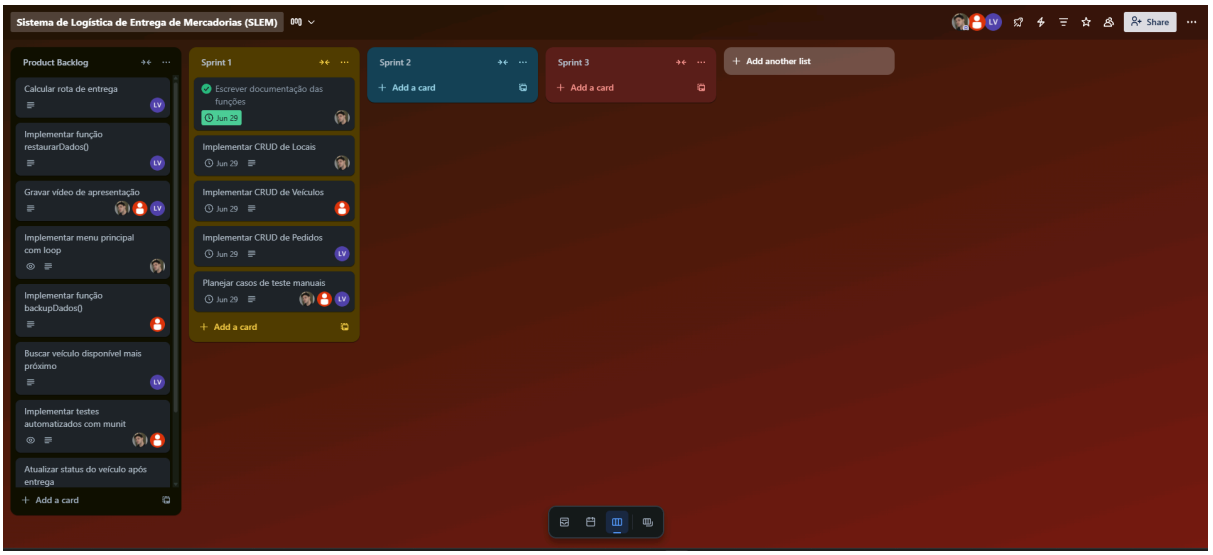


Backlog do Produto

SPRINT 1

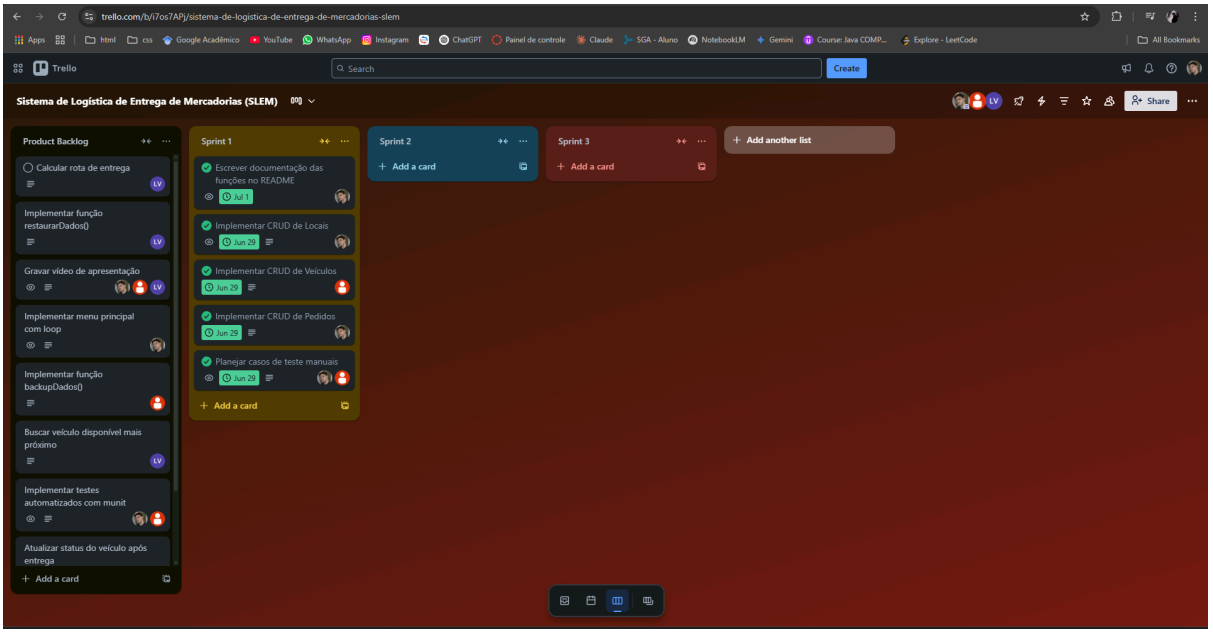
A imagem 1 apresenta o time criado no Trello com as divisões dos quadros. O quadro “Product Backlog” mostra as divisões das funções entre os integrantes e quais tarefas deverão ser entregues por cada integrante por ordem de sprints. Já o quadro “Sprint 1” mostra a organização inicial das funcionalidades que cada integrante ficou responsável de implementar até o final da primeira Sprint.

Imagem 1:



A imagem 2 apresenta como ficou o quadro “Sprint 1” na entrega das funcionalidades da primeira Sprint feitas pelos integrantes do grupo ao final da Sprint.

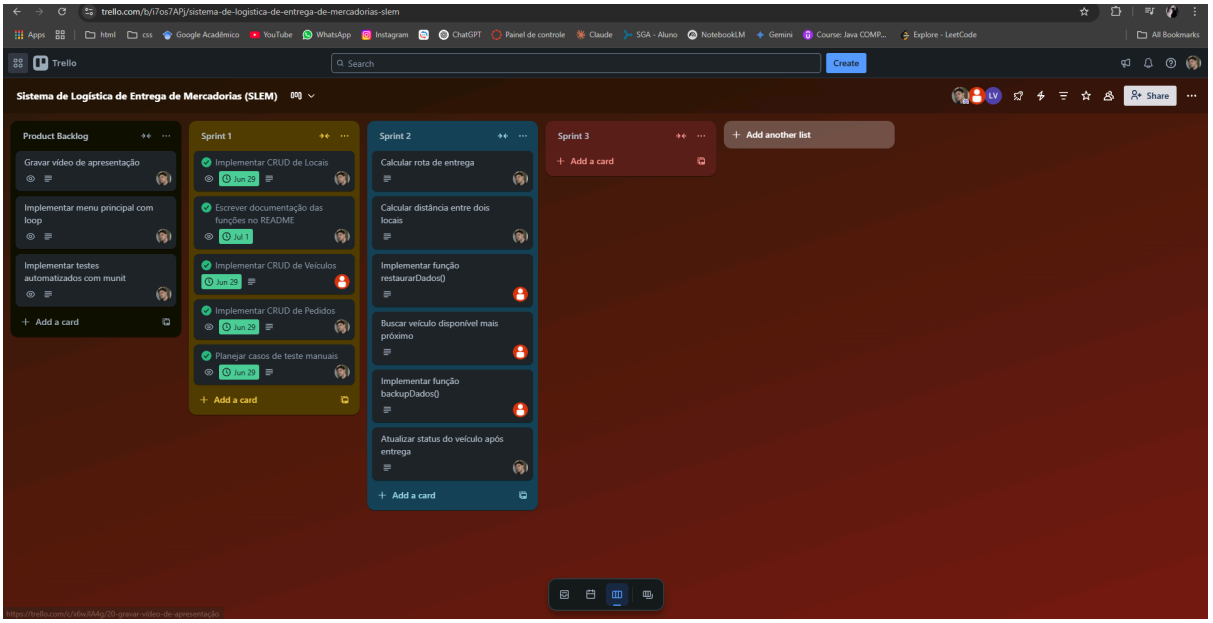
Imagem 2:



SPRINT 2

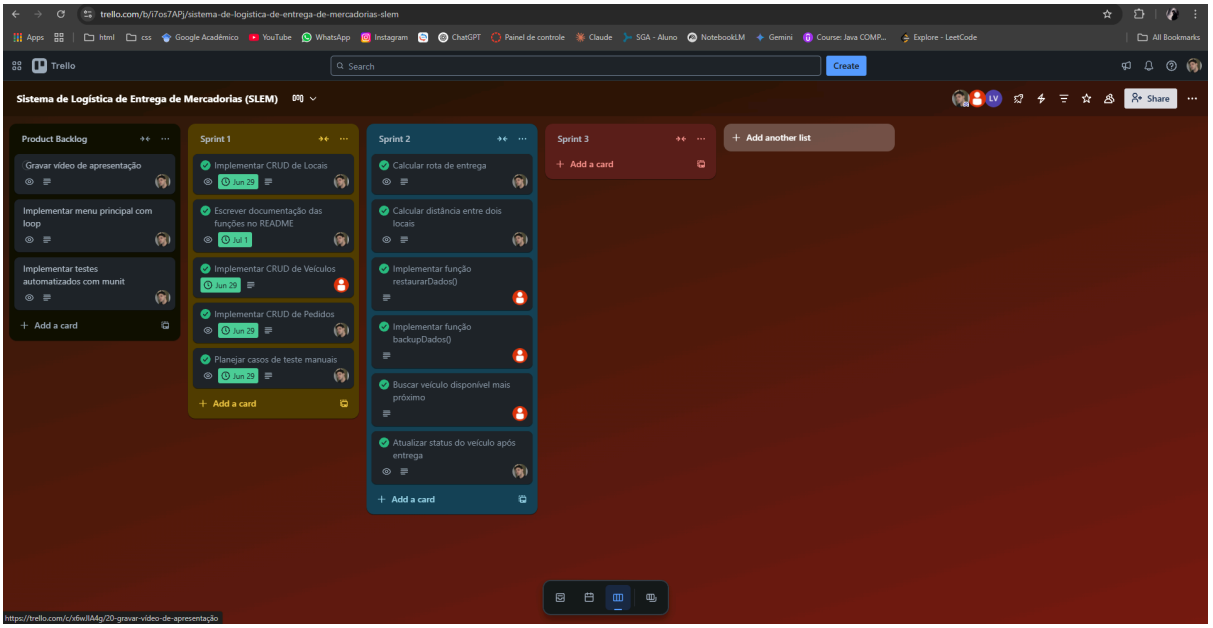
A imagem 3 apresenta o quadro “Sprint 2” para planejamento e organização das funcionalidades que cada integrante ficará responsável de desenvolver até o final da segunda Sprint.

Imagem 3:



A imagem 4 mostra como ficou o quadro “Sprint 2” na entrega das funcionalidades que foram designadas para a segunda Sprint do projeto.

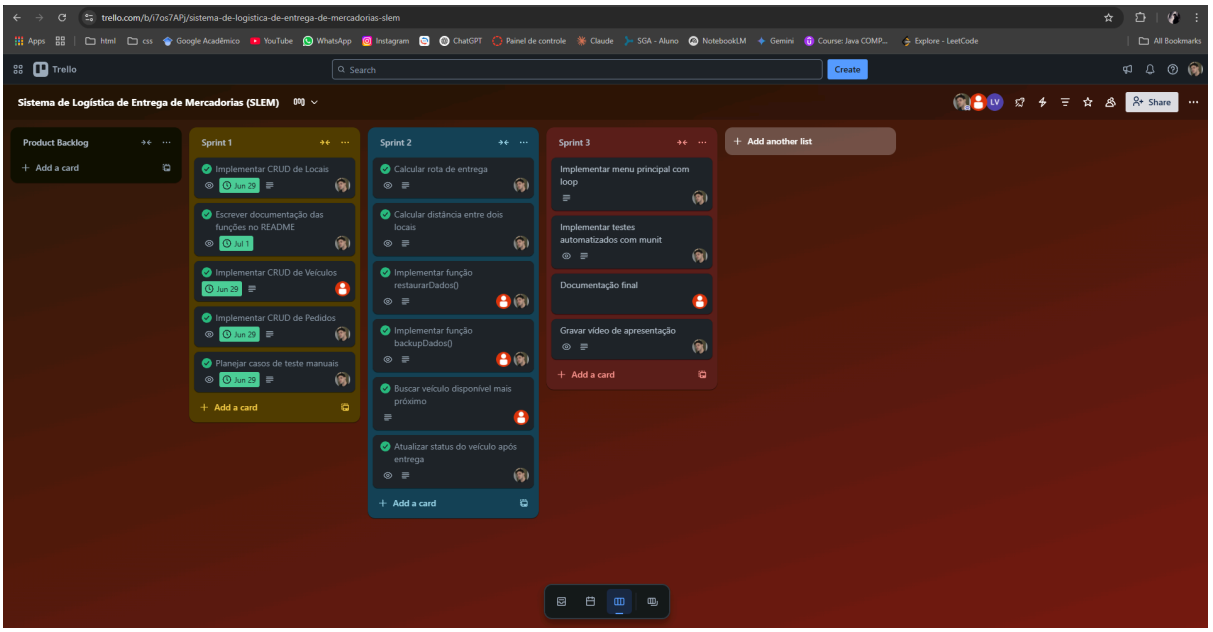
Imagem 4:



SPRINT 3

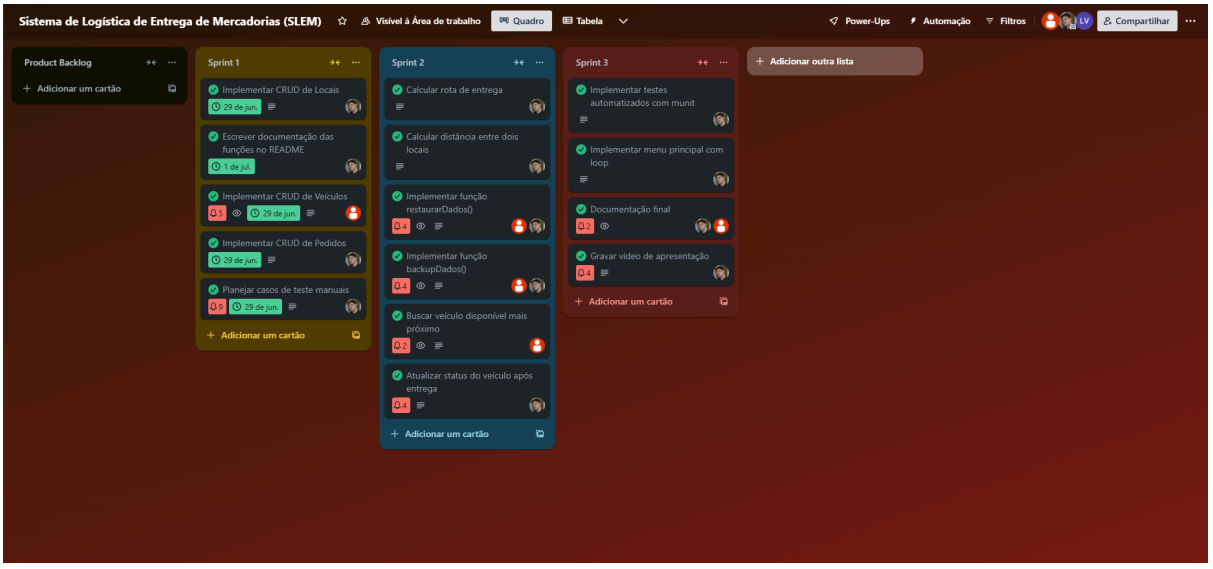
A imagem 5 apresenta o quadro “Sprint 3” onde ocorre o planejamento de funcionalidades, atividades de testes e atividades externas que deverão ser concluídas até o final da terceira Sprint.

Imagem 5:

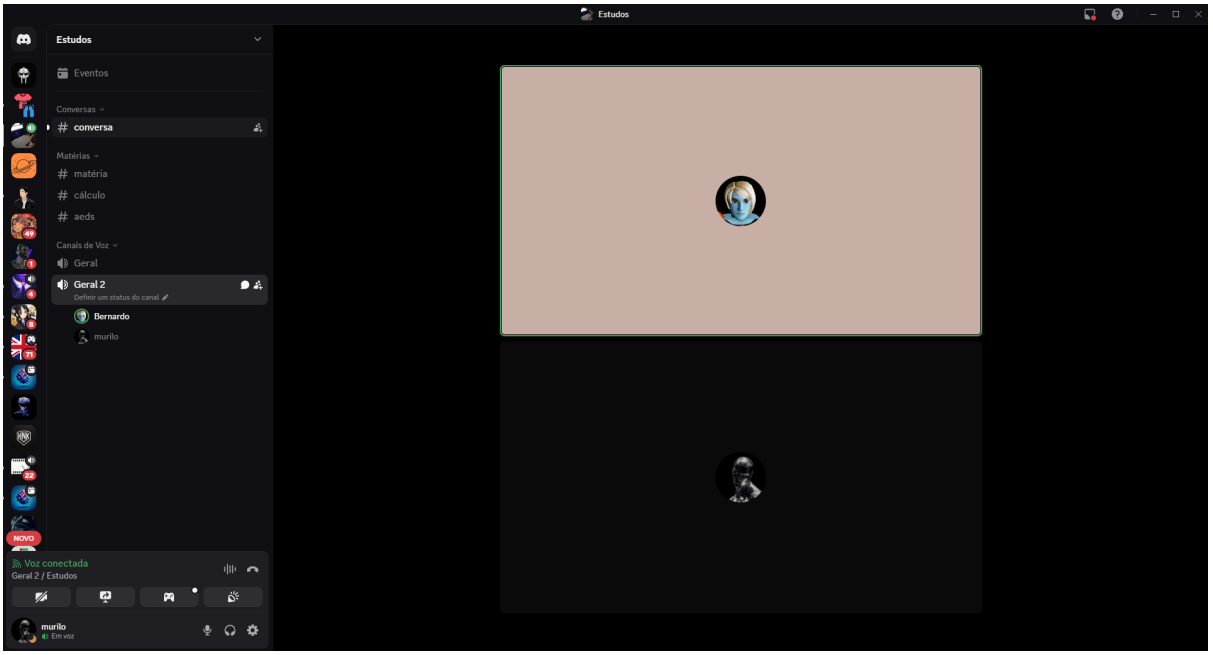


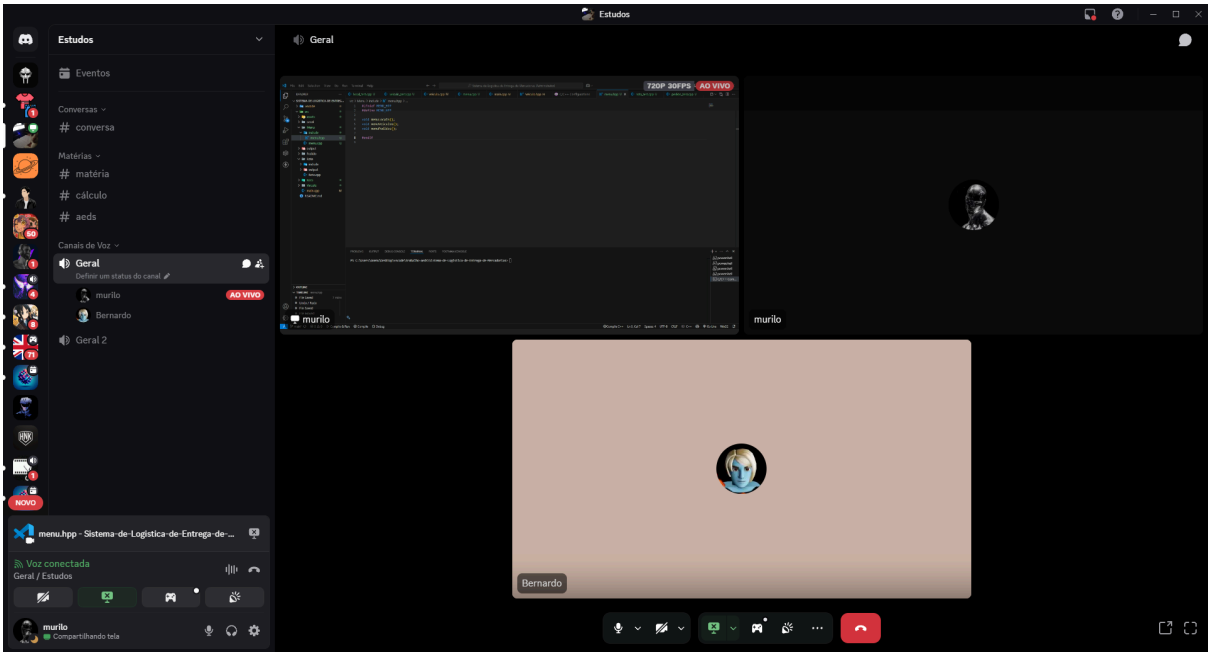
E a seguinte e última imagem apresenta como ficou o quadro “Sprint 3” ao final das entregas da última Sprint.

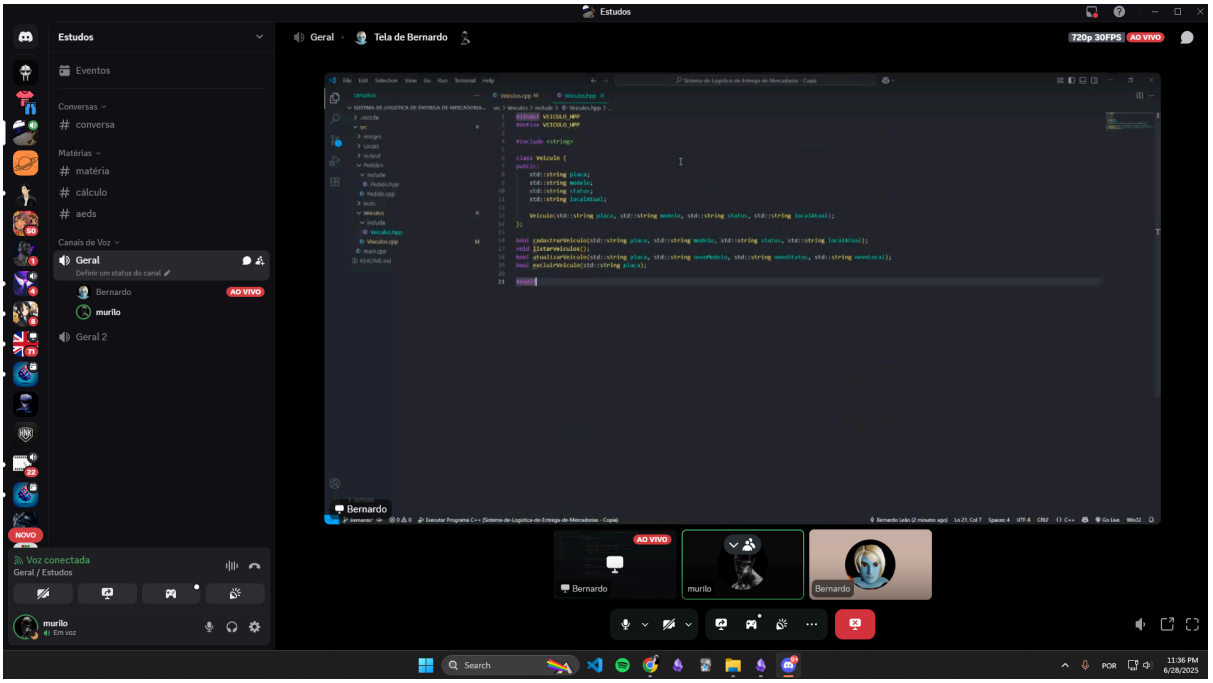
Imagem 6:



ALGUMAS PRINTS DAS REUNIÕES DAS 3 SRINTS FEITAS:







Lista de assinaturas das funções e parâmetros
A seguir estão listadas as principais funções utilizadas no sistema, com suas respectivas assinaturas e explicações dos parâmetros utilizados:

Módulo LOCAL

1.bool cadastrarLocal(const std::string& nome, float x, float y)

- Cadastra um novo local com nome e coordenadas X e Y.
- Parâmetros:
  - nome: nome do local.
  - x, y: coordenadas no espaço.
- Retorno: true se cadastrado com sucesso.

2.void listarLocais()

- Lista todos os locais cadastrados no sistema.
- Parâmetros: nenhum.

3.bool atualizarLocal(const std::string& nomeAntigo, const std::string& nomeNovo, float novoX, float novoY)

- Atualiza o nome e coordenadas de um local existente.
- Parâmetros:
    - nomeAntigo: nome atual do local.
    - nomeNovo: novo nome.
    - novoX, novoY: novas coordenadas.
- Retorno: true se atualizado com sucesso.

4.bool excluirLocal(const std::string& nome)

- Exclui um local pelo nome.
- Parâmetros:
    - nome: nome do local a ser removido.
- Retorno: true se excluído.

5.float calcularDistanciaEntreLocais(const Local& l1, const Local& l2)

- Calcula a distância entre dois objetos Local.

- Parâmetros:
    - `l1`, `l2`: objetos do tipo `Local`.
- Retorno: valor float da distância.

6.float calcularDistanciaEntreLocaisPorNome(const std::string& nome1, const std::string& nome2)

- Calcula a distância entre dois locais usando seus nomes.
- Parâmetros:
    - `nome1`, `nome2`: nomes dos locais.
- Retorno: valor float da distância.

7.Local* buscarLocalPorNome(const std::string& nome)

- Retorna um ponteiro para o local buscado.
- Parâmetros:
    - `nome`: nome do local.
- Retorno: ponteiro `Local*` (ou `nullptr` se não encontrado).

Módulo VEÍCULO

`1.bool cadastrarVeiculo(const std::string& placa, const std::string& modelo, const std::string& status, const std::string& localAtual)`

- Cadastra um novo veículo no sistema.
- Parâmetros:
    - `placa`: identificador do veículo.
    - `modelo`: nome do modelo.
    - `status`: estado atual (ex: "disponível", "em rota").
    - `localAtual`: nome do local onde o veículo se encontra.
- Retorno: `true` se cadastrado com sucesso.

`2.void listarVeiculos()`

- Exibe todos os veículos registrados no sistema.

`3.bool atualizarVeiculo(const std::string& placa, const std::string& novoModelo, const std::string& novoStatus, const std::string& novoLocal)`

- Atualiza os dados de um veículo específico.
- Parâmetros: placa atual, novo modelo, novo status e novo local.
- Retorno: `true` se a atualização for bem-sucedida.

4.bool excluirVeiculo(const std::string& placa)

- Remove um veículo do sistema com base na placa.
- Retorno: true se excluído com sucesso.

5.Veiculo* buscarVeiculoPorPlaca(const std::string& placa)

- Retorna um ponteiro para o veículo buscado.
- Retorno: ponteiro Veiculo* ou nullptr se não encontrado.

Módulo PEDIDO

1.bool cadastrarPedido(int codigo, const std::string& origem, const std::string& destino, float peso)

- Cadastra um novo pedido de entrega no sistema.
- Parâmetros:
  - codigo: identificador numérico único do pedido.
  - origem: local de retirada.
  - destino: local de entrega.
  - peso: peso da mercadoria.
- Retorno: true se cadastrado com sucesso.

2.void listarPedidos()

- Exibe todos os pedidos cadastrados.

3.bool atualizarPedido(int codigo, const std::string& novaOrigem, const std::string& novoDestino, float novoPeso)

- Atualiza os dados de um pedido existente.
- Retorno: true se atualizado.

4.bool excluirPedido(int codigo)

- Remove um pedido com base no seu código.

5.Pedido* buscarPedidoPorCodigo(int codigo)

- Busca um pedido e retorna ponteiro para ele, se encontrado.

Módulo ROTA

1.bool calcularRota(int codigoPedido, std::vector<std::string>& rotaCalculada)

- Calcula a melhor rota entre origem e destino de um pedido.
- **Parâmetros:**
  - codigoPedido: código do pedido que será roteado.
  - rotaCalculada: vetor de strings com a sequência dos locais.

2.void exibirRota(const std::vector<std::string>& rota)

- Exibe a rota calculada de forma legível.

Módulo BACKUPDADOS

1.void backupDados()

- Realiza o salvamento dos dados atuais do sistema em arquivos de backup.

Módulo RECUPERARDADOS

1.void restaurarDados()

- Restaura os dados do sistema a partir de backups previamente criados.

TESTES

Casos de Testes de Software

Entradas	Classes Válidas	Resultado Esperado	Classes Inválidas	Resultado Esperado
Local: Nome, X e Y	Nome não vazio e coordenadas válidas, ex: "Centro", 0.0, 0.0	Local cadastrado com sucesso	Nome vazio, coordenadas inválidas (ex: strings, negativos absurdos)	Cadastro rejeitado com mensagem de erro
Placa, Modelo, Status, Local do veículo	Placa: "ABC1234", Modelo: "Fiorino", Status: "disponivel", Local existente	Veículo cadastrado com sucesso	Placa vazia ou com caracteres inválidos, local inexistente	Cadastro rejeitado, solicitada correção dos dados
Pedido: Código, Origem, Destino, Peso	Código: 1, Origem ≠ Destino, Locais válidos, Peso > 0	Pedido cadastrado corretamente	Código negativo ou repetido, origem igual ao destino, peso negativo	Cadastro rejeitado e informado erro
Código de pedido para busca	Código existente: 1	Pedido encontrado, dados exibidos	Código inexistente: 9999	Pedido não encontrado, exibida mensagem de erro
Código de pedido para exclusão	Código existente	Pedido removido com sucesso	Código inexistente	Exclusão falha, exibida mensagem ao usuário
Código de pedido para cálculo de rota	Código existente com origem e destino cadastrados	Rota gerada e exibida corretamente	Código inexistente, origem/destino inexistentes	Rota não é gerada, exibida mensagem de erro
Executar backup de dados	Dados previamente cadastrados	Arquivos de backup salvos corretamente	Arquivos com falha de permissão	Mensagem de erro informando falha na gravação

Restaurar dados do backup	Arquivos válidos de backup disponíveis	Dados carregados com sucesso	Arquivos inexistentes ou corrompidos	Erro ao restaurar, sistema mantém dados atuais

Relatório de Execução de Testes

Menu principal do sistema

Entradas	Resultado	Aprovado?
Valor: 0	Encerra o programa com mensagem de finalização	Sim
Valor: 1	Inicia a função `menuLocais()` para gerenciar locais	Sim
Valor: 2	Inicia a função `menuVeiculos()` para gerenciar veículos	Sim
Valor: 3	Inicia a função `menuPedidos()` para gerenciar pedidos	Sim
Valor: 4	Inicia a função `backupDados()` para salvar os dados	Sim
Valor: 5	Inicia a função `restaurarDados()` para restaurar os dados	Sim
Valor: 6	Solicita uma nova entrada (opção inválida)	Sim

Valor: -1	Solicita uma nova entrada (opção inválida)	Sim
Valor: 'a'	Solicita uma nova entrada (entrada não numérica)	Sim

Menu de Local

Entradas	Resultado	Aprovado?
Valor: 1	Iniciada função de cadastro de local	Sim
Valor: 2	Iniciada função de listagem de locais	Sim
Valor: 3	Iniciada função de atualização de local	Sim
Valor: 4	Iniciada função de exclusão de local	Sim
Valor: 0	Retorna ao menu principal	Sim
Valor: 5	Entrada inválida, solicita nova opção	Sim
Valor: 'x'	Entrada inválida, solicita nova opção	Sim

Menu de Veículo

Entradas	Resultado	Aprovado?
Valor: 1	Iniciada função de cadastro de veículo	Sim
Valor: 2	Iniciada função de listagem de veículos	Sim
Valor: 3	Iniciada função de atualização de veículo	Sim
Valor: 4	Iniciada função de exclusão de veículo	Sim
Valor: 0	Retorna ao menu principal	Sim
Valor: -1	Entrada inválida, solicita nova opção	Sim

Menu de Pedido

Entradas	Resultado	Aprovado?
Valor: 1	Iniciada função de cadastro de pedido	Sim
Valor: 2	Iniciada função de listagem de pedidos	Sim
Valor: 3	Iniciada função de atualização de pedido	Sim
Valor: 4	Iniciada função de exclusão de pedido	Sim
Valor: 5	Iniciada função de busca de pedido	Sim
Valor: 0	Retorna ao menu principal	Sim
Valor: '7'	Entrada inválida, solicita nova opção	Sim

Menu de Rota

Entrada	Resultado Esperado	Resultado Obtido	Aprovado?
Valor: 1	Iniciar função de cadastro de rota	Iniciada função de cadastro de rota	Sim
Valor: 2	Iniciar função de listagem de rotas	Iniciada função de listagem de rotas	Sim
Valor: 3	Iniciar função de exclusão de rota	Iniciada função de exclusão de rota	Sim
Valor: 4	Iniciar função de busca de rota	Iniciada função de busca de rota	Sim
Valor: 5	Iniciar função de geração de rota otimizada	Iniciada função de geração de rota otimizada	Sim
Valor: 0	Retornar ao menu principal	Retorna ao menu principal	Sim
Valor: '-'	Entrada inválida, solicitar nova opção	Entrada inválida, solicita nova opção	Sim

Menu de Backup de Dados

Entrada	Resultado Esperado	Resultado Obtido	Aprovado ?
Valor: 1	Salvar dados de locais em arquivo	Dados de locais salvos com sucesso	Sim
Valor: 2	Carregar dados de locais a partir do arquivo	Dados de locais carregados corretamente	Sim
Valor: 3	Salvar dados de pedidos em arquivo	Dados de pedidos salvos com sucesso	Sim
Valor: 4	Carregar dados de pedidos a partir do arquivo	Dados de pedidos carregados corretamente	Sim
Valor: 5	Salvar dados de rotas em arquivo	Dados de rotas salvos com sucesso	Sim
Valor: 6	Carregar dados de rotas a partir do arquivo	Dados de rotas carregados corretamente	Sim
Valor: 0	Retornar ao menu principal	Sistema retorna ao menu principal	Sim
Valor: 'x'	Entrada inválida, solicitar nova opção	Solicita nova entrada ao usuário	Sim