Objetivo 1 — Escolha e definição do tema da aplicação

TEMA DO PROJETO
Desenvolvimento de um Sistema de Gestão de Transporte por Aplicativo

Descrição do Sistema
O projeto consiste na criação de um software para gerenciar corridas de transporte privado, funcionando de forma parecida com o Uber. O foco principal é organizar o fluxo entre passageiros que precisam se deslocar e motoristas que prestam o serviço. O sistema permite cadastrar os usuários, os veículos disponíveis e registrar as viagens, calculando automaticamente o custo do trajeto e mantendo os dados salvos de forma estruturada.

Problema a ser Resolvido
Atualmente, a gestão de transporte feita de forma improvisada ou manual traz diversos riscos e dificuldades:

- Insegurança: Sem um registro digital, é impossível rastrear quem foi o motorista ou o veículo em uma determinada viagem.
- Erros Financeiros: Calcular o valor das corridas manualmente gera inconsistências nos preços, o que pode prejudicar tanto o passageiro quanto o lucro do motorista.
- Falta de Histórico: A dificuldade em consultar viagens passadas impede um controle financeiro eficiente e a resolução de eventuais problemas relatados pelos usuários.

O sistema proposto automatiza esses processos, garantindo que as informações sejam precisas e fáceis de consultar.

Principais Entidades do Sistema

- Passageiro: Classe que armazena os dados pessoais de quem solicita as viagens.
- Motorista: Classe que contém as informações do profissional, incluindo sua CNH.
- Veículo: Registro do automóvel (modelo, placa e cor) que será utilizado nas corridas.
- Viagem: É o registro central que une o passageiro ao motorista e ao veículo, calculando a tarifa final.

Funcionalidades Básicas

- Cadastro e edição de passageiros e motoristas.
- Vínculo de veículos aos motoristas cadastrados.
- Simulação de chamadas de viagens com definição de destino.
- Cálculo automático do valor da corrida.
- Listagem de histórico de todas as viagens realizadas no sistema.

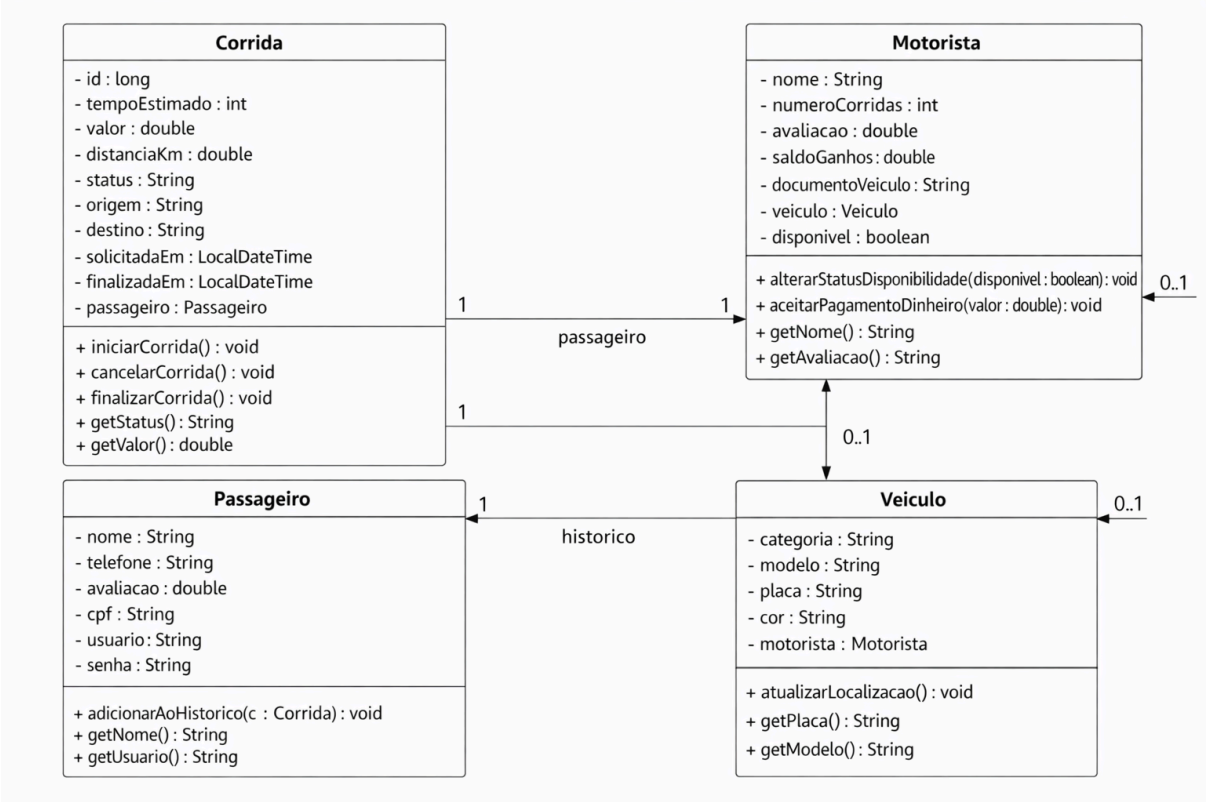
Tecnologias Utilizadas

- Linguagem: Java.
- Interface: Java Swing (Interface Gráfica).
- Arquitetura: Padrão MVC (Model-View-Controller).
- Documentação: Diagrama de Classes UML.
- Versionamento: GitHub.

Justificativa da Escolha do Tema

Escolhi este tema por ser um exemplo muito comum no nosso cotidiano, o que facilita a compreensão da lógica do software. Do ponto de vista técnico, o sistema de transporte é excelente para aplicar os conceitos de Programação Orientada a Objetos (POO) que estamos estudando, como a interação entre diferentes classes e o encapsulamento de dados. É um projeto prático que simula bem os desafios reais de desenvolvimento de software e organização de banco de dados.

Objetivo 2 – Modelagem do sistema com UML



Relacionamentos:

- **Passageiro → Corrida: Quem solicita e paga pela viagem.**
- **Motorista → Corrida: Quem aceita e realiza o serviço.**
- **Veículo → Motorista: Quem opera e está vinculado ao recurso físico (carro).**

Objetivo 5 — Apresentação da aplicação desenvolvida

Introdução do Sistema

O sistema desenvolvido é uma **aplicação desktop em Java**, utilizando a biblioteca **Swing**, criada para **simular um sistema básico de transporte por aplicativo**.

O sistema permite:

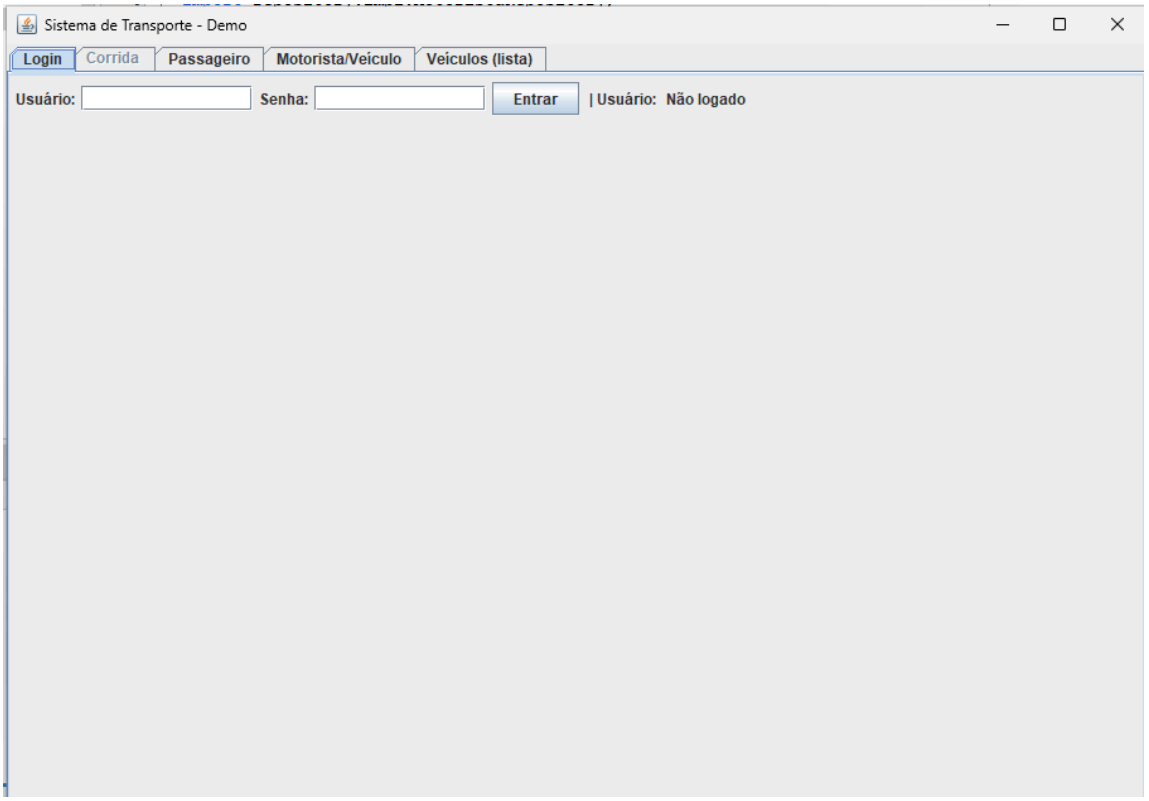
- **Login de usuário com usuário e senha**
- **Cadastro e gerenciamento de passageiros**
- **Cadastro e gerenciamento de motoristas**
- **Cadastro e gerenciamento de veículos**
- **Solicitação de corridas**
- **Cálculo automático do valor da corrida**
- **Finalização e cancelamento de corridas**
- **Visualização do resumo (recibo) da corrida**
- **Consulta e listagem de informações do sistema**

O sistema foi organizado seguindo o padrão **MVC (Model–View–Controller)**, separando claramente:

- **As regras de negócio** (Models e Controllers)
- **A interface gráfica** (Views em Swing)
- **A persistência de dados simulada**, utilizando **repositories em memória**

Essa organização facilita a manutenção, a compreensão do código e a evolução futura do sistema.

Tela Principal – Ponto Inicial do Sistema



O sistema sempre inicia pela **Tela Principal**, que funciona como o menu central da aplicação.

Nela, o usuário encontra:

- **Aba de Login**
- **Aba de Corrida**
- **Acesso aos Cadastros**
- **Acesso às Listagens do Sistema**

O sistema foi estruturado a partir de uma tela principal feita com um `JFrame`, que funciona como a base da aplicação.”

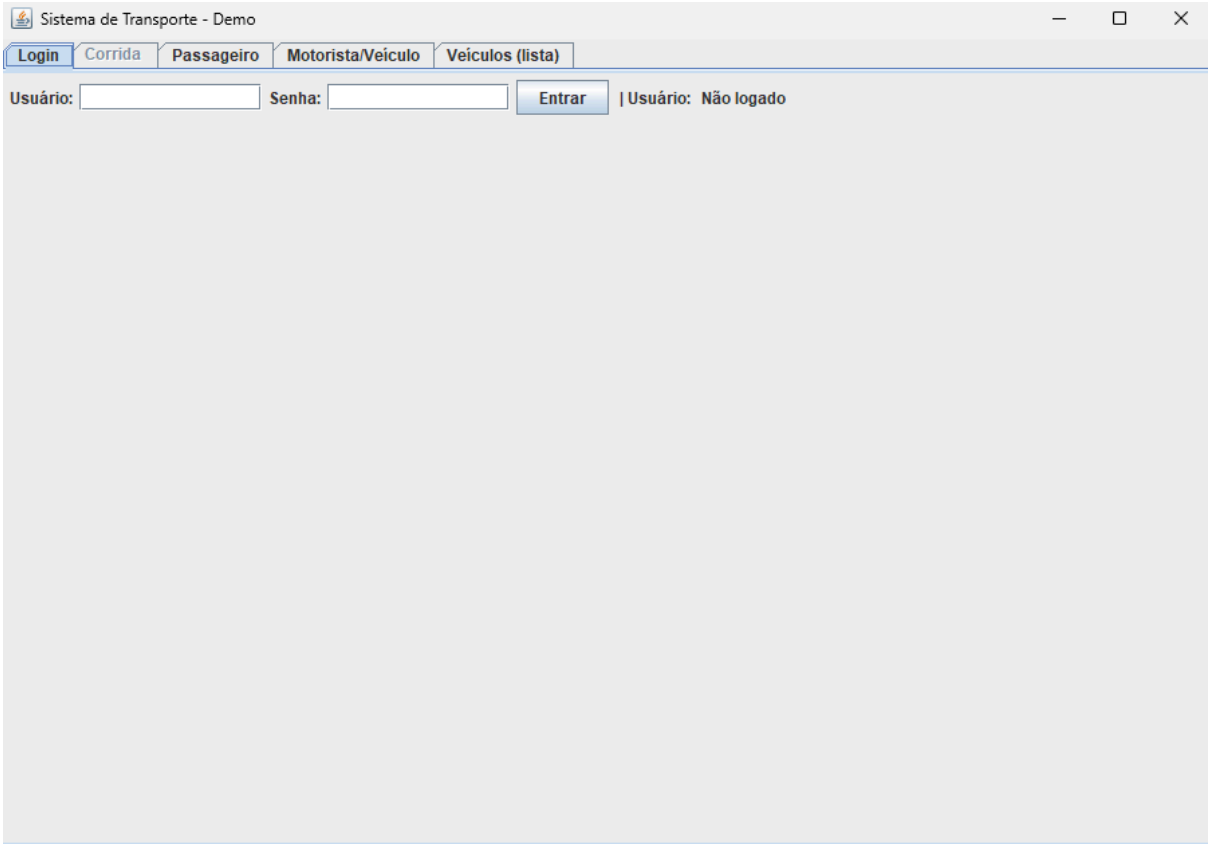
Eu usei `JTabbedPane` para organizar o sistema em abas:

Login, Corrida, Passageiro, Motorista/Veículo, Listagens

Por que isso foi feito assim:

Essa maneira facilita a navegação, centraliza todas as funcionalidades em uma única janela e evita a abertura de várias telas separadas.

Login do Sistema



O sistema possui uma tela de login, que exige:

- **Usuário**
- **Senha**

Somente após a autenticação bem-sucedida o usuário consegue:

- Acessar a aba **Corrida**
- Solicitar uma corrida
- Utilizar as funcionalidades principais do sistema

A tela de login foi criada usando componentes básicos do Java Swing.

Componentes usados:

- `JTextField` → usuário
- `JPasswordField` → senha
- `JButton` → entrar

- `JLabel` → status do login

Lógica aplicada:

Quando o botão de login é clicado, o sistema verifica se o usuário e a senha informados estão entre os passageiros cadastrados.

- Se estiver correto:

  O usuário é marcado como **logado**

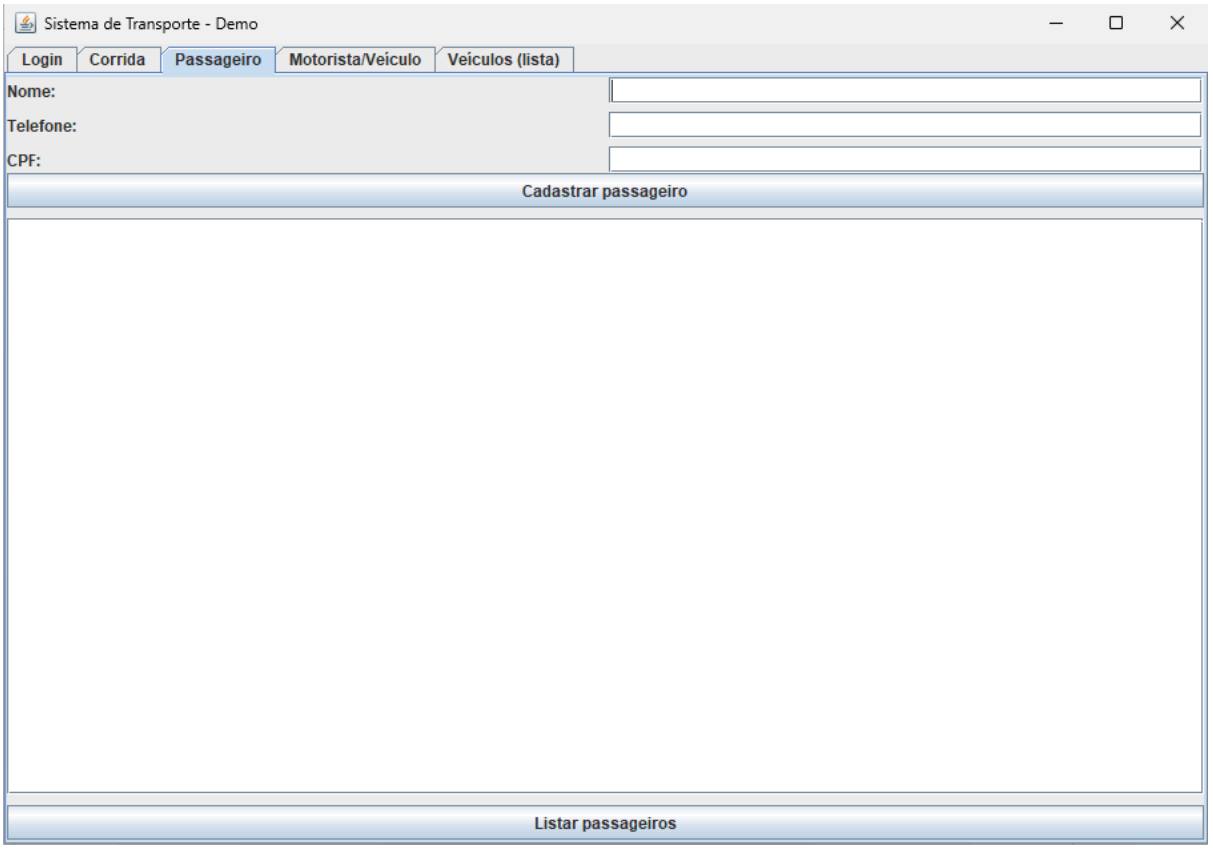
  O CPF do usuário fica armazenado para uso nas corridas

- Se não:

  O sistema bloqueia o acesso às funcionalidades principais

O sistema possui os seguintes cadastros principais:

Cadastro de Passageiro



Como eu fiz:

Essa tela foi desenvolvida para registrar os passageiros do sistema, utilizando campos de entrada para dados pessoais e de autenticação.

Componentes usados:

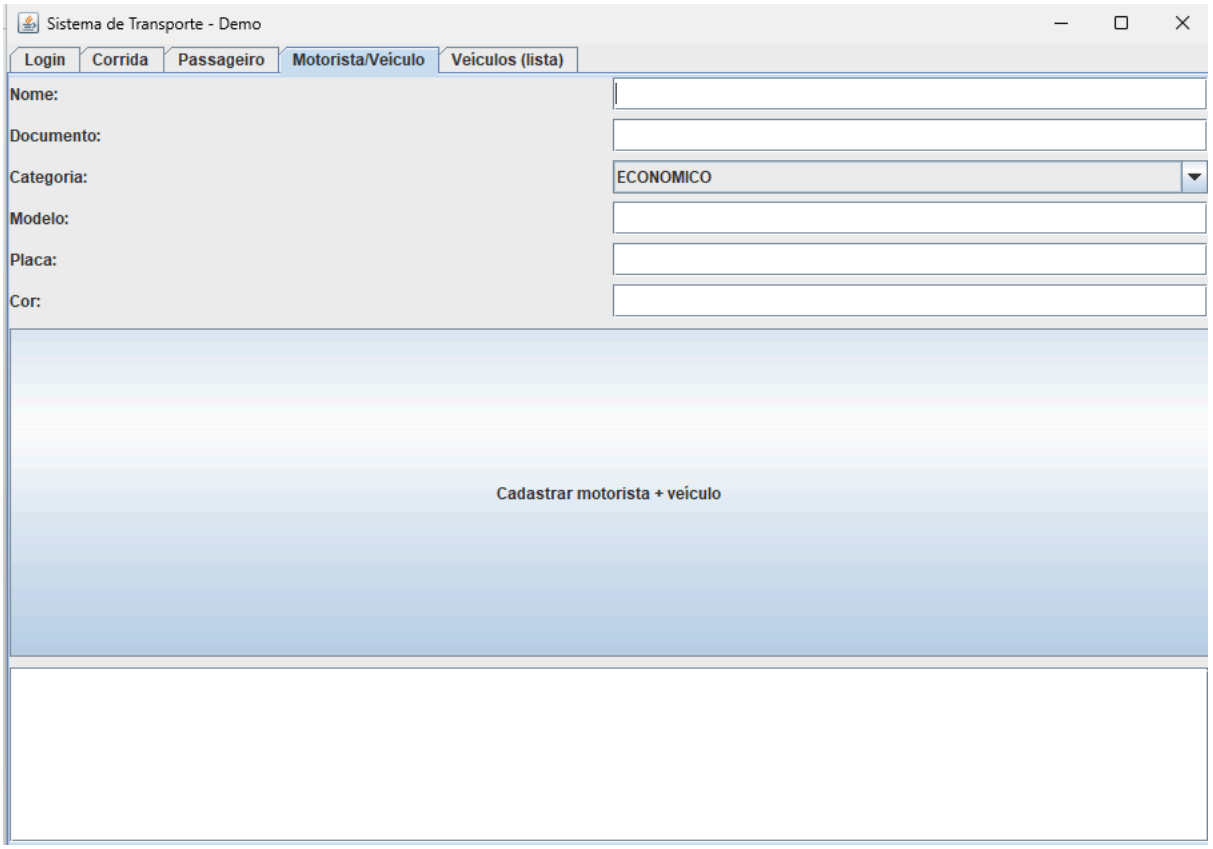
- `JTextField` para nome, telefone, CPF e usuário
- `JPasswordField` para senha
- `JButton` para cadastrar
- Área de texto ou lista para exibir passageiros cadastrados

O que acontece ao cadastrar:

Ao clicar no botão de cadastro, os dados são capturados dos campos e usados para criar um objeto Passageiro.

- Esse objeto é:
    - Armazenado em uma **lista ou repositório**
    - Utilizado depois para:
        - Login
        - Solicitação de corridas
        - Histórico de viagens

Cadastro de Motorista e Veículo



Como eu fiz:

> Nessa tela o cadastro do motorista e do veículo é feito de forma integrada, ou seja, os dois são criados juntos.

Componentes usados:

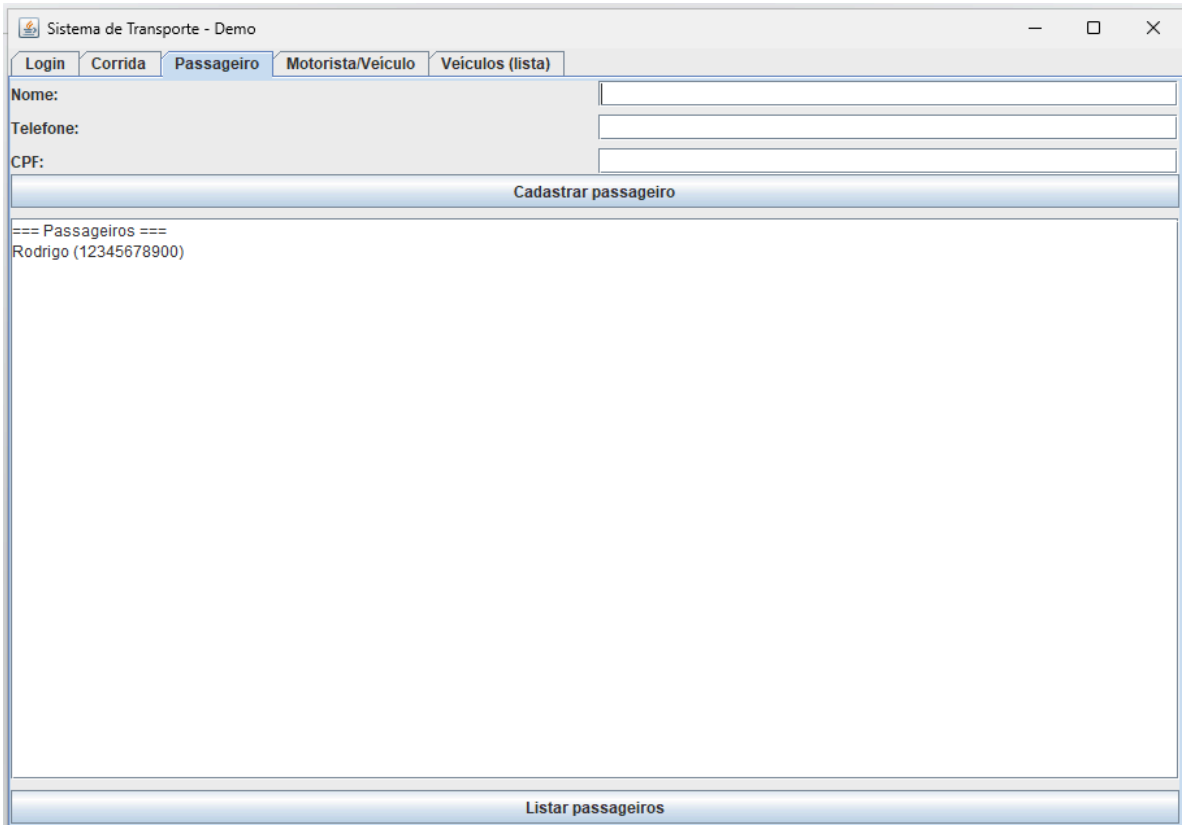
- `JTextField` → dados do motorista e do veículo
- `JComboBox` → categoria do veículo (Econômico, etc.)
- `JButton` → cadastrar

Lógica aplicada:

> "Ao cadastrar, o sistema cria um objeto Motorista e associa a ele um objeto Veículo."

- Isso garante que:
  - Todo motorista tenha um veículo
  - Toda corrida use um veículo válido
- A categoria do veículo é usada depois no cálculo da corrida

Listagens e Consultas do Sistema



Como eu fiz:

> As telas de listagem foram criadas apenas para consulta dos dados armazenados no sistema.

Componentes usados:

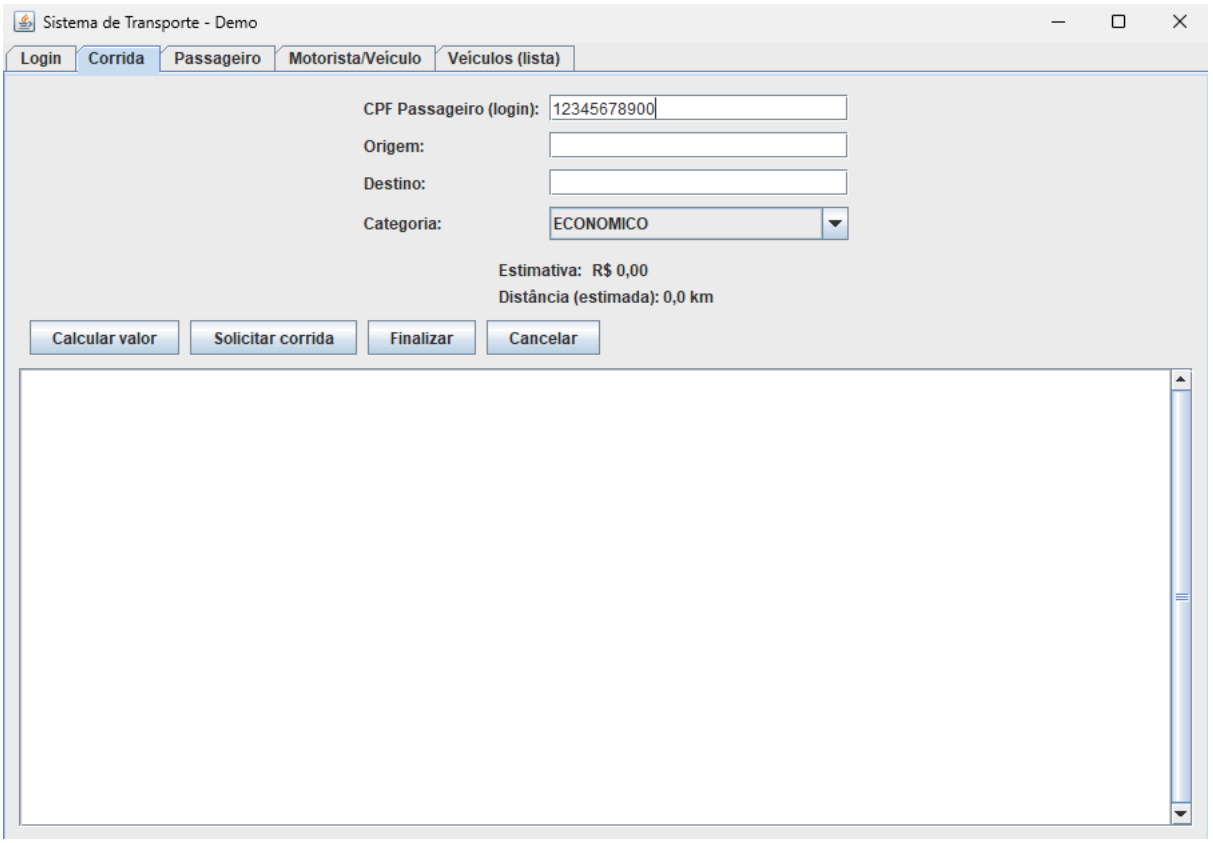
- `JTextArea`, `JList` ou `JTable`
- `JButton` para atualizar/listar

Funcionamento:

> Essas telas apenas percorrem as listas de passageiros, motoristas, veículos ou corridas e exibem os dados formatados.

- Não alteram dados
- Não fazem cadastro
- Apenas leem informações do sistema

Solicitação de Corrida – Fluxo Principal do Sistema



Como eu fiz:

> Essa tela representa o fluxo principal do sistema e só pode ser usada por um passageiro logado.

Componentes usados:

- `JTextField` → origem e destino
- `JComboBox` → categoria do veículo
- `JButton` → calcular valor, solicitar, finalizar e cancelar

Lógica da corrida:

> O sistema usa a categoria do veículo e a distância estimada para calcular o valor da corrida.

- Cria um objeto Corrida
- Associa:
  - Passageiro logado
  - Motorista disponível
  - Veículo compatível
- Registra a corrida no sistema

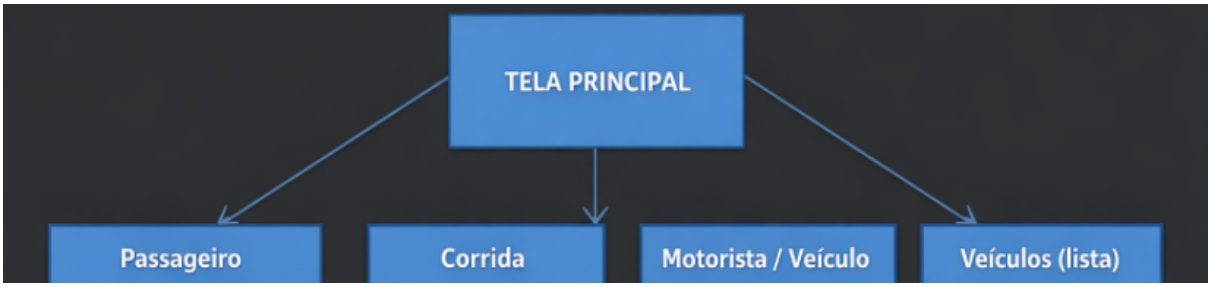
Relacionamento das Entidades

A corrida relaciona diretamente:

- **Passageiro**
- **Motorista**
- **Veículo**

Esse relacionamento é representado no **diagrama UML do sistema**, garantindo uma estrutura organizada e coerente com o funcionamento do sistema.

Fluxo Geral do Sistema



Conclusão

O sistema desenvolvido apresenta uma interface gráfica clara, intuitiva e bem organizada, na qual cada tela possui uma função específica dentro do fluxo da aplicação. Essa estrutura garante uma navegação eficiente para o usuário, facilitando a compreensão e o uso correto do sistema. Além disso, foi adotada uma padronização visual entre as telas, o que contribui para um aprendizado mais rápido e uma melhor experiência de uso.

A Tela Principal atua como o ponto central de navegação do sistema, permitindo que o usuário acesse de forma simples e rápida as principais funcionalidades, como o cadastro de passageiros, motoristas e veículos, a gestão das corridas e a consulta de registros. Esse modelo centralizado proporciona um fluxo lógico de utilização, evitando confusões e tornando o uso do sistema mais objetivo e organizado.

As telas de cadastro seguem um padrão visual consistente, possibilitando incluir, editar, remover e visualizar informações de passageiros, motoristas e veículos. Essa padronização facilita o gerenciamento dos dados, garante maior controle sobre as informações cadastradas e reduz a ocorrência de erros operacionais durante a utilização do sistema.

A tela de Corrida integra diferentes dados do sistema, reunindo informações de passageiros, motoristas e veículos em um único processo. Nessa etapa, o usuário pode acompanhar os dados da viagem, simular o trajeto e visualizar o cálculo automático do valor da corrida, tornando o funcionamento do sistema mais próximo da realidade de um aplicativo de transporte. Essa integração evidencia a aplicação prática dos conceitos de Programação Orientada a Objetos e da arquitetura MVC.

De modo geral, o sistema apresenta uma boa organização das telas, um fluxo de navegação bem definido e uma interface funcional, atendendo aos objetivos propostos inicialmente. O projeto demonstra de forma clara a aplicação prática dos conceitos estudados ao longo do desenvolvimento, resultando em uma solução didática, funcional e preparada para futuras melhorias e expansões.