UNIVERSIDADE DE BRASÍLIA

# **TRABALHO FINAL:** SISTEMA DE MOBILIDADE URBANA

MATEUS FERNANDES DANTAS – 251026248

MATHEUS VINÍCIUS - 251026257

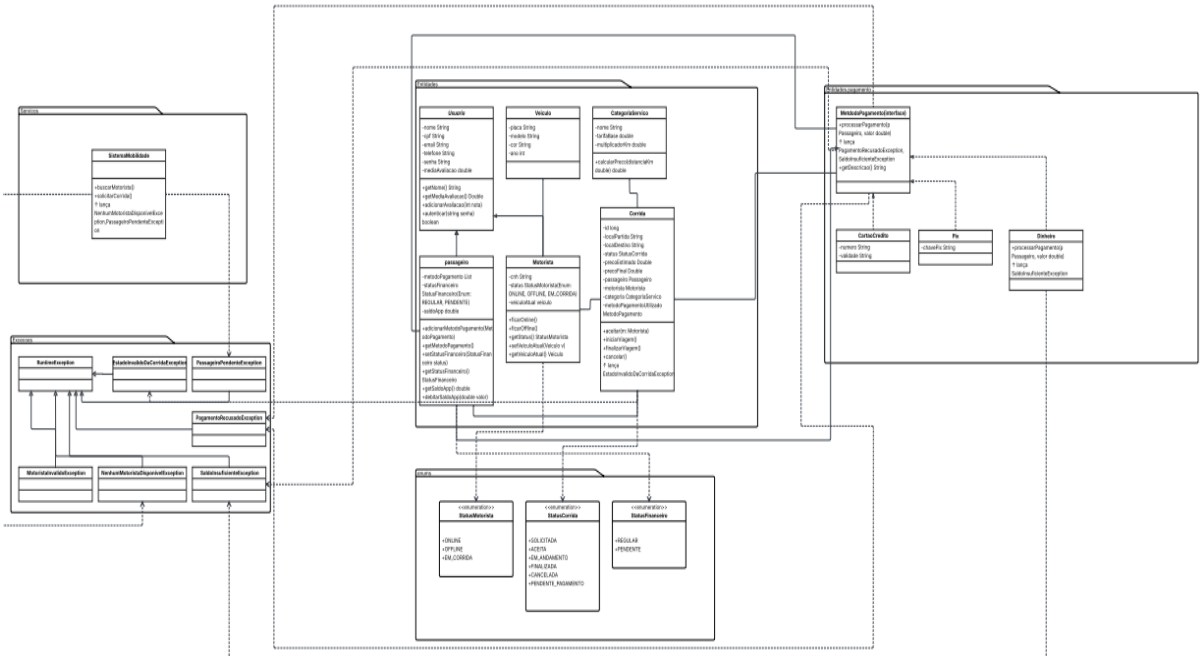
HYBSON MURILO CORRENTE - 242028833

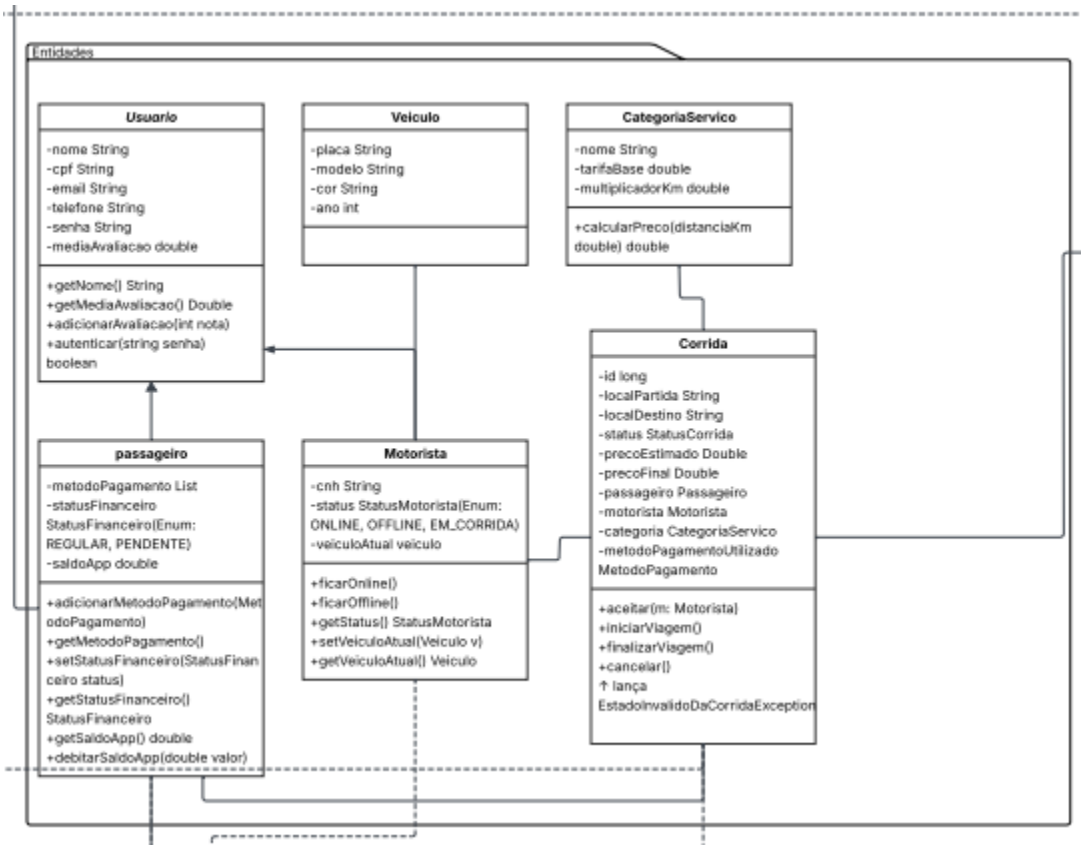
RICARDO MEDEIROS REZENDE - 242028762

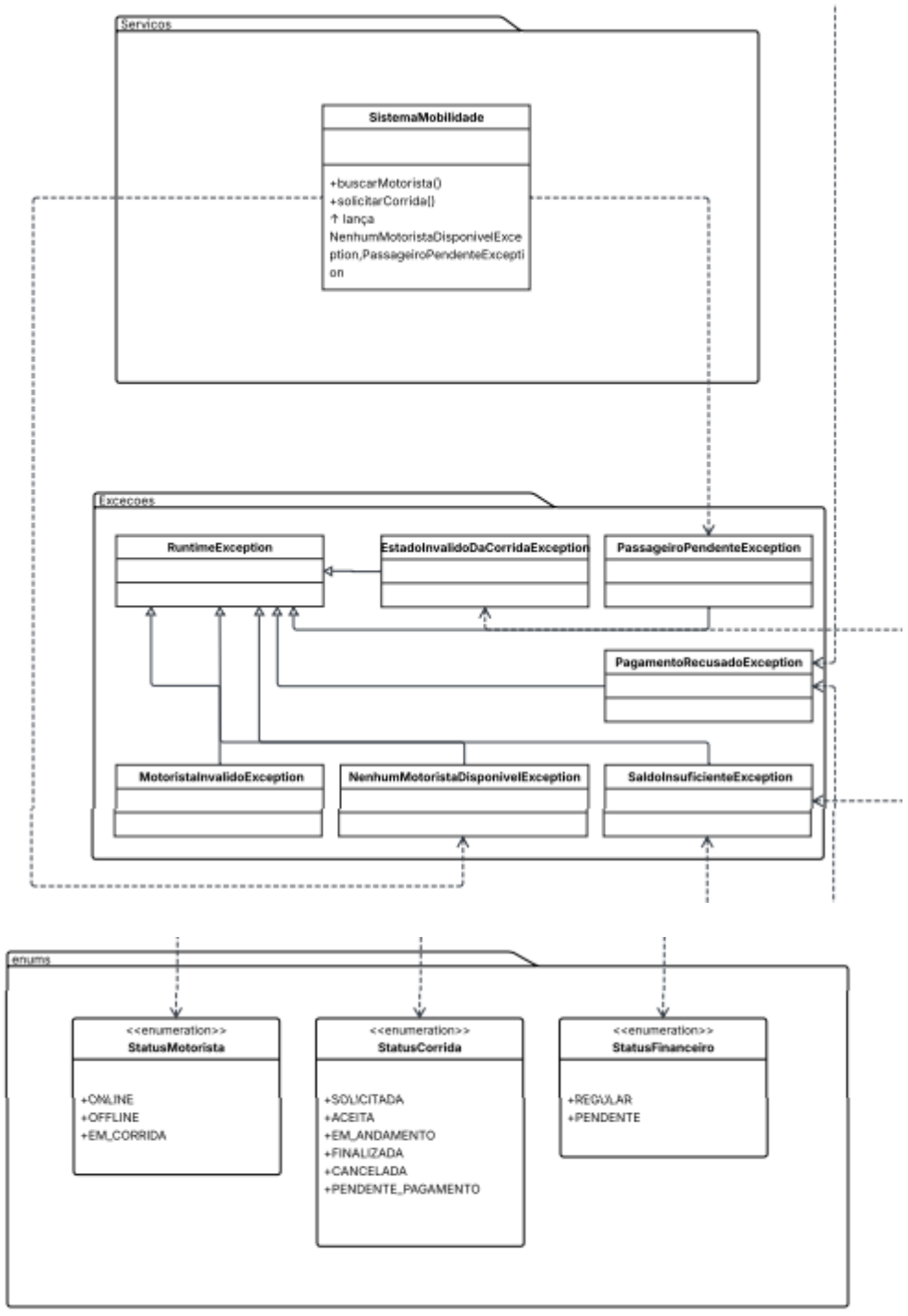
GAMA

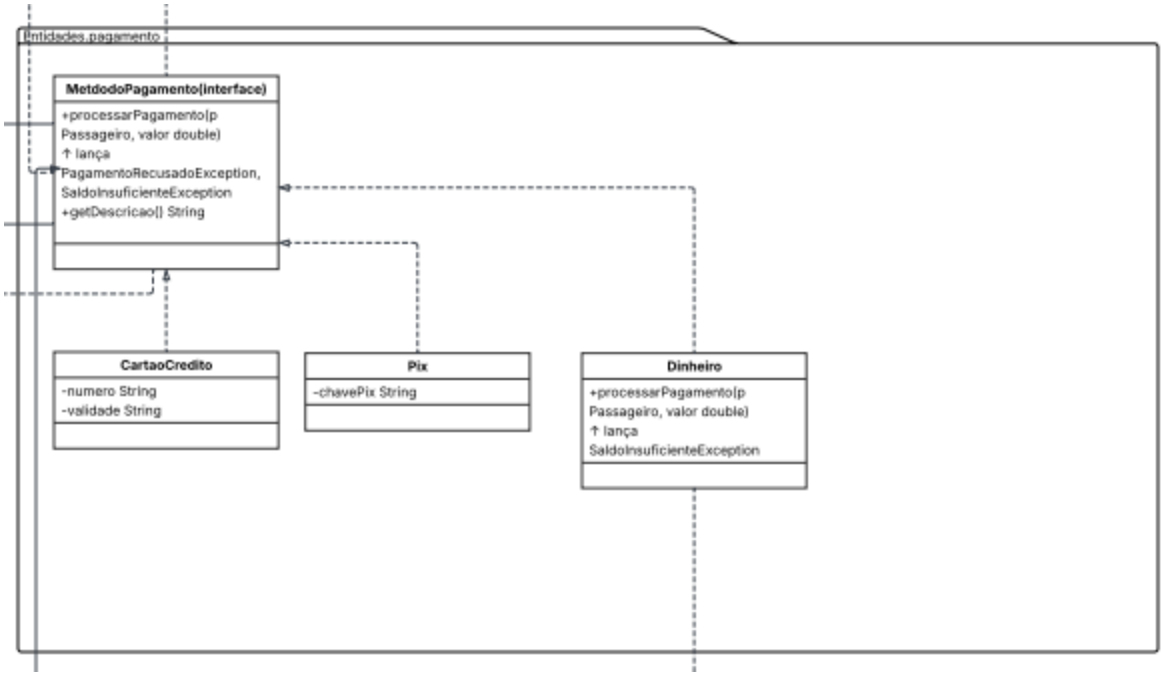
2025

## 1. DIAGRAMA DE CLASSES UML.






Servicos
SistemaMobilidade
+buscarMotorista()
+solicitarCorrida()
↑ lança
NenhumMotoristaDisponivelExce
ption,PassageiroPendenteExcepti
on
Excecoes
RuntimeException
EstadoInvalidoDaCorridaException
PassageiroPendenteException
PagamentoRecusadoException
MotoristaInvalidoException
NenhumMotoristaDisponivelException
SaldoInsuficienteException
enums
<<enumeration>>
StatusMotorista
+ONLINE
+OFFLINE
+EM_CORRIDA
<<enumeration>>
StatusCorrida
+SOLICITADA
+ACEITA
+EM_ANDAMENTO
+FINALIZADA
+CANCELADA
+PENDENTE_PAGAMENTO
<<enumeration>>
StatusFinanceiro
+REGULAR
+PENDENTE




https://lucid.app/lucidchart/9cbd26f4-c9f7-4bb0-8e4b-8f1885a28588/edit?invitationId=inv_7db5d5e0-4615-40b7-bbc0-6211b3e284a5&page=0_0#

**2.Explicação das associações, heranças e polimorfismo aplicados.**

A arquitetura do sistema foi desenhada e planejada, para garantir um baixo acoplamento e alta coesão, utilizando todos os conceitos cobrados em Orientação a Objetos:

### 2.1 Herança

A herança foi aplicada para facilitar a generalização dos atributos e criar uma hierarquia semântica:

**Entidades:** As classes Passageiro e Motorista herdam da superclasse Usuario, permitindo assim centralizar atributos comuns como: Nome, Email e Telefone, facilitando a manutenção do código.

**Exceções:** Todas as exceções personalizadas do pacote excecoes herdam da classe Exception ou RuntimeException, aproveitando a estrutura robusta de tratamento de erros nativa da linguagem Java.

### 2.2 Associação e Composição

As classes de domínio interagem através de associações diretas:

**Corrida:** A classe Corrida atua como a entidade centralizadora, pois ela agrega instâncias de Passageiro, Motorista e CategoriaServiço.

Além disso também temos a relação entre Motorista e Veículo que demonstra uma importante associação para o funcionamento do código, pois todo motorista possui um veículo ativo para realizar o trabalho.

### 2.3 Polimorfismo e Interfaces

O pacote pagamento demonstra um uso de polimorfismo via interfaces:

A interface MetodoPagamento define o contrato processarPagamento()

**Corrida:** A classe Corrida não depende de implementações concretas como os métodos de pagamento Pix ou Dinheiro, mas sim da abstração da interface, permitindo assim trocar a forma de pagamento em tempo de execução sem alterar a lógica da corrida, respeitando o princípio de estar aberto para extensões, mas fechado para modificações.

## 3.Justificativa para as Exceções Customizadas

Optamos pela criação de um pacote dedicado as exceções contendo classes específicas de erro, ao invés de utilizar exceções genéricas, os motivos são:

- **Clareza do uso:** As exceções como por exemplo NenhumMotoristaDisponivelException descreve exatamente o seu uso e qual foi a violação cometida pelo usuário, isso torna o código autoexplicativo e facilita o trabalho de outros desenvolvedores.
- **Obrigatoriedade de tratamento:** Ao herdar de Exception, tornamos erros críticos em *Checked Exceptions*, por exemplo o compilador obriga que quem chame o método finalizarViagem() trate o erro explicitamente através do try-catch, isso garante que o sistema não quebre e que o usuário receba um feedback adequado.
- **Controle de Fluxo Específico:** Permite capturar erros de forma linear, o sistema pode reagir de maneiras diferentes para cada exceção colocada, como por exemplo para um SaldoInsuficienteException que pede para adicionar fundos e para um EstadoInvalidodaCorridaException que indica um erro lógico grave, impedindo ações ilegais como cancelar uma viagem já finalizada.