Análise e Projeto Orientado a Objetos

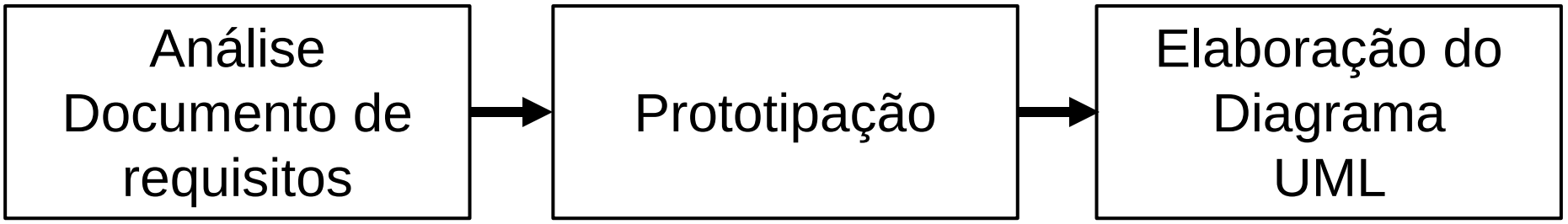
Prof. Flores
jose.flores@unicesumar.edu.br

Vamos estudar:

- Digrama de Classes (Exemplo Prático)

Diagrama de Classes (Exercício)

Fases



Análise do Documento de Requisitos

Diagrama de Classes (Exercício)

Documento de Requisitos

1. Introdução

O sistema de locadora de carros é projetado para gerenciar o aluguel de veículos por clientes. Ele permite que funcionários registrem aluguéis, clientes realizem reservas, e o sistema controle pagamentos, disponibilidade de veículos e categorias de preços.

2. Objetivos

- Facilitar o processo de aluguel de veículos.
- Garantir a disponibilidade de informações sobre veículos e clientes.
- Automatizar cálculos de valores de aluguel e processamento de pagamentos.
- Gerar relatórios sobre aluguéis e veículos disponíveis.

Diagrama de Classes (Exercício)

3. Requisitos Funcionais

1. **Gerenciamento de Clientes**
   - Cadastrar novos clientes.
   - Atualizar informações de clientes existentes.
   - Excluir clientes inativos.
2. **Gerenciamento de Veículos**
   - Cadastrar novos veículos.
   - Atualizar informações de veículos (ex.: disponibilidade).
   - Excluir veículos fora de uso.
3. **Aluguel de Veículos**
   - Permitir que clientes reservem veículos.
   - Calcular o valor total do aluguel com base na categoria e duração.
   - Finalizar aluguéis após a devolução do veículo.
4. **Pagamentos**
   - Processar pagamentos de aluguéis.
   - Registrar métodos de pagamento (cartão, dinheiro, etc.).
5. **Relatórios**
   - Gerar relatórios de aluguéis ativos e finalizados.
   - Listar veículos disponíveis e indisponíveis.

Diagrama de Classes (Exercício)

4. Requisitos Não Funcionais

1. **Desempenho**
   - O sistema deve ser capaz de processar consultas e operações em tempo hábil.
2. **Segurança**
   - As informações dos clientes e funcionários devem ser protegidas contra acesso não autorizado.
3. **Usabilidade**
   - A interface deve ser intuitiva para facilitar o uso por funcionários e clientes.
4. **Escalabilidade**
   - O sistema deve suportar o aumento do número de clientes e veículos sem perda de desempenho.

Diagrama de Classes (Exercício)

5. Casos de Uso Principais

1. **Cliente solicita aluguel de veículo**
   - Cliente escolhe um veículo disponível.
   - Funcionário registra o aluguel e calcula o valor.
   - Pagamento é processado.
2. **Funcionário gera relatório**
   - Funcionário acessa o sistema para visualizar relatórios de aluguéis e veículos.
3. **Cliente devolve veículo**
   - Funcionário atualiza o status do veículo e finaliza o aluguel.

Prototipação

Diagrama de Classes (Exercício)

Estudo das classes

Diagrama de Classes (Exercício)

Descrição das Classes

1. Cliente

- Representa um cliente da locadora.
- **Atributos** :
  - `id: int` - Identificador único do cliente.
  - `nome: String` - Nome completo do cliente.
  - `cpf: String` - CPF do cliente (formato "XXX.XXX.XXX-XX").
  - `telefone: String` - Número de telefone de contato.
  - `email: String` - Endereço de e-mail do cliente.
- **Métodos** :
  - `alugarVeiculo(veiculo: Veiculo): void`
    - Parâmetro: `veiculo` - O veículo que o cliente deseja alugar.
    - Retorno: `void` .
    - Descrição: Permite ao cliente solicitar um veículo em aluguel.
  - `devolverVeiculo(veiculo: Veiculo): void`
    - Parâmetro: `veiculo` - O veículo que o cliente está devolvendo.
    - Retorno: `void` .
    - Descrição: Processa a devolução de um veículo.

Diagrama de Classes (Exercício)

2. Funcionário

- Representa os funcionários responsáveis por gerenciar a locadora.
- **Atributos** :
  - `id: int` - Identificador único do funcionário.
  - `nome: String` - Nome completo do funcionário.
  - `cargo: String` - Cargo ocupado pelo funcionário (ex.: atendente, gerente).
  - `salario: double` - Salário mensal do funcionário.
- **Métodos** :
  - `registrarAluguel(cliente: Cliente, veiculo: Veiculo, dataInicio: Date, dataFim: Date): Aluguel`
    - Parâmetros:
      - `cliente: Cliente` - Cliente que está realizando o aluguel.
      - `veiculo: Veiculo` - Veículo que está sendo alugado.
      - `dataInicio: Date` - Data de início do aluguel.
      - `dataFim: Date` - Data de término do aluguel.
    - Retorno: `Aluguel` - Retorna o objeto `Aluguel` criado.
    - Descrição: Registra um novo aluguel no sistema.
  - `gerarRelatorio(): List<Aluguel>`
    - Parâmetros: Nenhum.
    - Retorno: `List<Aluguel>` - Retorna uma lista de todos os aluguéis registrados.
    - Descrição: Gera relatórios de aluguéis realizados.

Diagrama de Classes (Exercício)

3. Aluguel

- Representa um contrato de aluguel entre o cliente e a locadora.

- **Atributos** :
  - `id: int` - Identificador único do aluguel.
  - `dataInicio: Date` - Data de início do aluguel.
  - `dataFim: Date` - Data de término do aluguel.
  - `valorTotal: double` - Valor total do aluguel.
  - `status: String` - Status atual do aluguel (ex.: "ativo", "finalizado").
  - `cliente: Cliente` - Referência ao cliente que realizou o aluguel.
  - `veiculo: Veiculo` - Referência ao veículo alugado.
  - `pagamentos: List<Pagamento>` - Lista de pagamentos associados ao aluguel.

Diagrama de Classes (Exercício)

- **Métodos :**
  - `calcularValor(categoria: Categoria): double`
    - Parâmetro: `categoria: Categoria` - Categoria do veículo alugado.
    - Retorno: `double` - Retorna o valor total calculado.
    - Descrição: Calcula o valor total com base na duração e categoria do veículo.
  - `finalizarAluguel(): void`
    - Parâmetros: Nenhum.
    - Retorno: `void` .
    - Descrição: Finaliza o contrato de aluguel.
  - `adicionarPagamento(pagamento: Pagamento): void`
    - Parâmetro: `pagamento: Pagamento` - Pagamento a ser adicionado ao aluguel.
    - Retorno: `void` .
    - Descrição: Adiciona um pagamento à lista de pagamentos do aluguel.

Diagrama de Classes (Exercício)

4. Veículo

- Representa os veículos disponíveis para aluguel.
- **Atributos** :
  - `placa: String` - Placa do veículo (formato "XXX-XXXX").
  - `modelo: String` - Modelo do veículo.
  - `marca: String` - Marca do veículo.
  - `ano: int` - Ano de fabricação.
  - `disponivel: boolean` - Indica se o veículo está disponível para aluguel.
- **Métodos** :
  - `reservar(): void`
    - Parâmetros: Nenhum.
    - Retorno: `void` .
    - Descrição: Reserva o veículo para um cliente.
  - `liberar(): void`
    - Parâmetros: Nenhum.
    - Retorno: `void` .
    - Descrição: Libera o veículo após a devolução.

Diagrama de Classes (Exercício)

5. Pagamento

- Representa os pagamentos realizados pelos clientes.
- **Atributos** :
  - `id: int` - Identificador único do pagamento.
  - `valorPago: double` - Valor pago pelo cliente.
  - `metodoPagamento: String` - Método de pagamento utilizado (ex.: cartão, dinheiro).
- **Métodos** :
  - `processarPagamento(): boolean`
    - Parâmetros: Nenhum.
    - Retorno: `boolean` - Retorna `true` se o pagamento for processado com sucesso, caso contrário, `false` .
    - Descrição: Processa o pagamento do aluguel.

Diagrama de Classes (Exercício)

6. Categoria

- Representa as categorias dos veículos (ex.: econômico, luxo, SUV).
- **Atributos** :
  - `id: int` - Identificador único da categoria.
  - `nome: String` - Nome da categoria.
  - `precoDiaria: double` - Preço diário para aluguel de veículos dessa categoria.
- **Métodos** :
  - `calcularPreco(dias: int): double`
    - Parâmetro: `dias: int` - Número de dias de aluguel.
    - Retorno: `double` - Retorna o preço total calculado.
    - Descrição: Calcula o preço com base na categoria e número de dias.

Elaboração do Diagrama UML

Diagrama de Classes (Exercício)

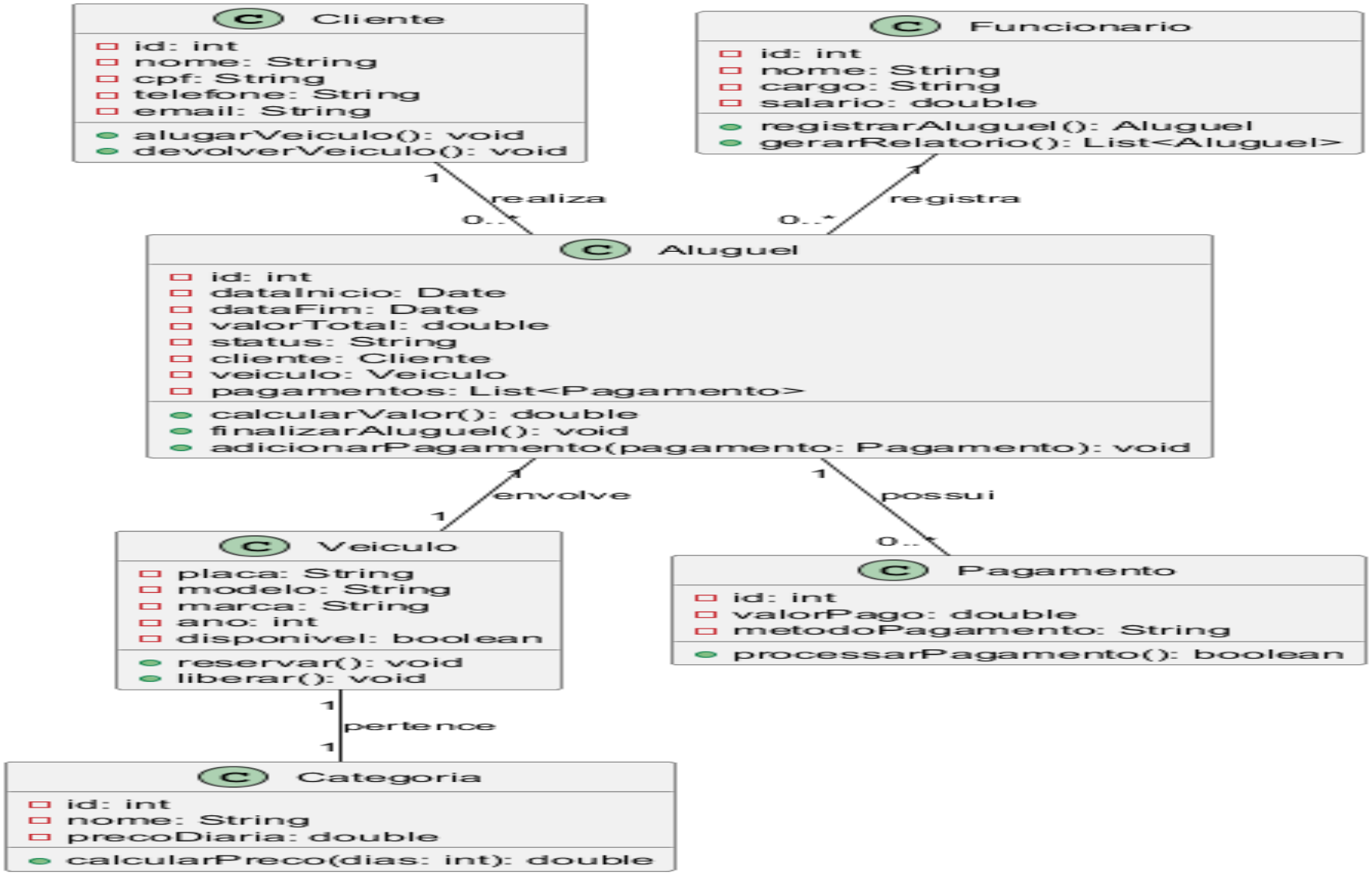


Diagrama de Classes (Exercício)

Explicação dos Relacionamentos

1. **Cliente-Aluguel**
   - Um cliente pode realizar vários aluguéis (**1:N**).
   - Um aluguel pertence a um único cliente (**1:1**).
2. **Aluguel-Veículo**
   - Um aluguel está associado a um único veículo (**1:1**).
   - Um veículo pode estar em vários aluguéis ao longo do tempo (**1:N**).
3. **Veículo-Categoria**
   - Um veículo pertence a uma única categoria (**1:1**).
   - Uma categoria pode conter vários veículos (**1:N**).
4. **Aluguel-Pagamento**
   - Um aluguel pode ter um ou mais pagamentos associados (**1:N**).
5. **Funcionário-Aluguel**
   - Um funcionário pode registrar vários aluguéis (**1:N**).
   - Um aluguel é registrado por um único funcionário (**1:1**).

Referências bibliográficas

Engenharia de software: uma abordagem profissional / Roger S. Pressman; tradução: Francisco Araújo da Costa.
https://research.ebsco.com/linkprocessor/plink?id=04e1c0fc-43b2-3c4a-ba49-86c4b9749234

Engenharia de software / Ian Sommerville; tradução: Luiz Levy Siqueira.
https://research.ebsco.com/linkprocessor/plink?id=e037bf66-82cb-330f-a5b2-294c550cea39

Princípios de análise e projeto de sistemas com UML / Eduardo Bezerra.
https://research.ebsco.com/linkprocessor/plink?id=3fb30a0e-8825-31a3-bdca-3dedaddd7979

Análise e projeto orientado a objetos / Autores: Déverson Rogério Rando... [et al.]; organizador: Simone Erbs da Costa; UniCesumar. Núcleo de Educação a Distância. Disponível em: https://bibliotecaweb.unicesumar.edu.br/pergamumweb/downloadArquivo?

UML 2: uma abordagem prática. / Autor: Gilleanes T. A. Guedes; Novatec. 2011: