

## Trabalho Prático - Fase 2

---

# Programação Orientada a Objetos

**Docente: Luís Gonzaga Martins Ferreira**

**Aluno: Rodrigo Lopes Ferreira Nº 31525**

Repositório GitHub:

[https://github.com/LFtech6/OOP\\_Project](https://github.com/LFtech6/OOP_Project)

## Índice

<b>Índice de Figuras .....</b>	<b>2</b>
<b>Resumo .....</b>	<b>3</b>
<b>1. Introdução.....</b>	<b>4</b>
<b>2. Enquadramento Teórico.....</b>	<b>4</b>
2.1 Programação Orientada a Objetos.....	4
2.2 Aplicação ao domínio .....	4
<b>3. Análise e Modelação do Sistema (Fase 1) .....</b>	<b>4</b>
3.1 Análise do Domínio .....	4
3.2 Modelação de Classes .....	5
3.3 Diagrama UML.....	5
<b>4. Implementação (Fase 1 e Fase 2).....</b>	<b>6</b>
4.1 Arquitetura em Camadas .....	6
4.2 Business Objects (BO) .....	6
4.3 Camada de Dados (DL) .....	6
4.4 Regras de Negócio (RN) .....	6
4.5 Gestão de Exceções .....	6
4.6 Testes Unitários.....	7
<b>5. Diagrama UML final.....</b>	<b>7</b>
<b>7. Conclusão.....</b>	<b>8</b>
<b>8 - Referências .....</b>	<b>9</b>

## Índice de Figuras

Figura 1 - Diagrama UML.....	5
Figura 2 – Diagrama UML final .....	7

## Resumo

Este relatório apresenta o desenvolvimento de um sistema de **Gestão de Condomínios**, realizado em duas fases no âmbito da unidade curricular de Programação Orientada a Objetos. Na Fase 1 foi efetuada a análise do domínio, a modelação das entidades e a implementação base seguindo os princípios da **POO**. Na Fase 2 o sistema foi evoluído através da organização em camadas (BO, DL, RN), persistência em ficheiros, validações de regras de negócio e gestão controlada de exceções. O trabalho resultante demonstra uma arquitetura simples, coerente e preparada para evolução futura.

# 1. Introdução

A gestão de condomínios envolve um conjunto alargado de entidades e processos, como moradores, frações, despesas, quotas, reuniões e documentos. A informatização deste tipo de sistemas permite organizar informação, reduzir erros e facilitar a tomada de decisão.

O objetivo deste trabalho foi desenvolver um sistema orientado a objetos que representasse este domínio de forma clara e estruturada. A metodologia seguida consistiu numa primeira fase de análise e modelação, seguida de uma segunda fase de implementação com separação de responsabilidades, persistência de dados e validação de regras de negócio.

## 2. Enquadramento Teórico

### 2.1 Programação Orientada a Objetos

A **POO** baseia-se na modelação de sistemas através de objetos que representam entidades do mundo real, combinando dados e comportamento. Os princípios de encapsulamento, abstração e reutilização permitem construir sistemas modulares e fáceis de manter.

### 2.2 Aplicação ao domínio

No contexto da gestão de condomínios, a POO facilita a representação de conceitos como morador, fração ou despesa, permitindo uma correspondência direta entre o modelo conceptual e a implementação em código.

## 3. Análise e Modelação do Sistema (Fase 1)

### 3.1 Análise do Domínio

Foram identificadas as principais entidades do sistema: Condomínio, Morador (Proprietário e Inquilino), Fração, Despesa, Quota, Pagamento, Reunião e Documento. As relações entre estas entidades refletem o funcionamento real de um condomínio.

### 3.2 Modelação de Classes

As entidades foram representadas por classes, privilegiando a clareza e a separação de responsabilidades. A herança foi utilizada para modelar diferentes tipos de moradores, evitando duplicação de código.

### 3.3 Diagrama UML

O diagrama UML sintetiza as classes principais e as respetivas relações, servindo como base para a implementação.

A Fase 1 concluiu com a:

- definição da arquitetura essencial do sistema;
- implementação de todas as classes com atributos e propriedades;
- documentação e organização do código;
- preparação lógica para etapas posteriores (e.g., cálculo real, validações, coleções dinâmicas).

Esta base serve como alicerce para a Fase 2, onde serão implementados comportamentos e relações dinâmicas entre entidades.

Segue aqui um anexo do diagrama UML que representa as minhas classes:

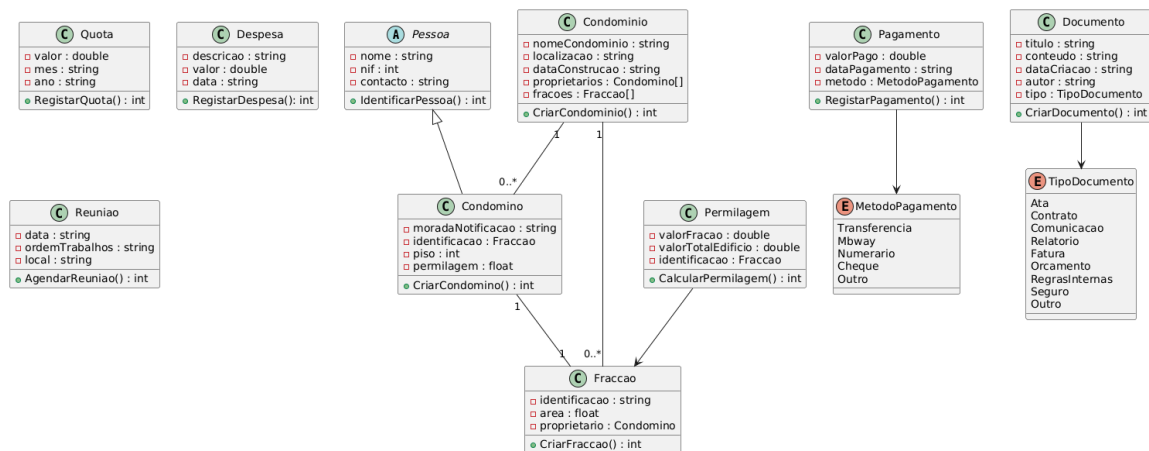


Figura 1 - Diagrama UML

## 4. Implementação (Fase 1 e Fase 2)

### 4.1 Arquitetura em Camadas

Na Fase 2 o sistema foi reorganizado em camadas:

- **BO (Business Objects)** – representam apenas dados do domínio;
- **RN (Regras de Negócio)** – validam e calculam regras do sistema;
- **DL (Camada de Dados)** – gere coleções e persistência em ficheiros;
- **Exceções** – tratamento específico de erros de sistema.

### 4.2 Business Objects (BO)

Os BO contêm apenas atributos e propriedades, sem validações nem acesso a dados externos. Exemplos incluem Morador, Fração, Quota, Pagamento e Documento.

### 4.3 Camada de Dados (DL)

A persistência é realizada através de serialização binária para ficheiros. Cada entidade relevante possui uma classe DL responsável pelas operações CRUD e pela leitura/escrita em ficheiro.

### 4.4 Regras de Negócio (RN)

As RN concentram validações e cálculos, como:

- validação de quotas (mês, ano, valores);
- cálculo da permissão;
- cálculo do valor da quota.

As RN não lançam exceções, devolvendo resultados booleanos e mensagens de erro, mantendo o controlo explícito do fluxo do programa.

### 4.5 Gestão de Exceções

Foi implementada apenas uma exceção personalizada (FicheiroException), utilizada na camada de dados para erros críticos de acesso a ficheiros. Esta abordagem simplifica o sistema e facilita a explicação do modelo.

#### 4.6 Testes Unitários

Com o objetivo de garantir a correção e fiabilidade de parte da solução desenvolvida, foram implementados **testes unitários** recorrendo ao *framework* **MSTest**, incidindo sobre a camada de **Dados**, mais concretamente sobre a gestão de moradores.

Dado o âmbito do trabalho e o tempo disponível, optou-se por concentrar os testes unitários apenas na classe Moradores, assegurando a verificação das principais operações CRUD básicas, nomeadamente:

- Inserção de moradores na estrutura interna
- Limpeza da lista de moradores
- Persistência de dados em ficheiro
- Leitura dos dados previamente guardados

## 5. Diagrama UML final

Após a conclusão do desenvolvimento das funcionalidades previstas para as **Fases 1 e 2**, foi elaborado o **modelo UML final**, refletindo de forma fiel a estrutura definitiva do sistema.

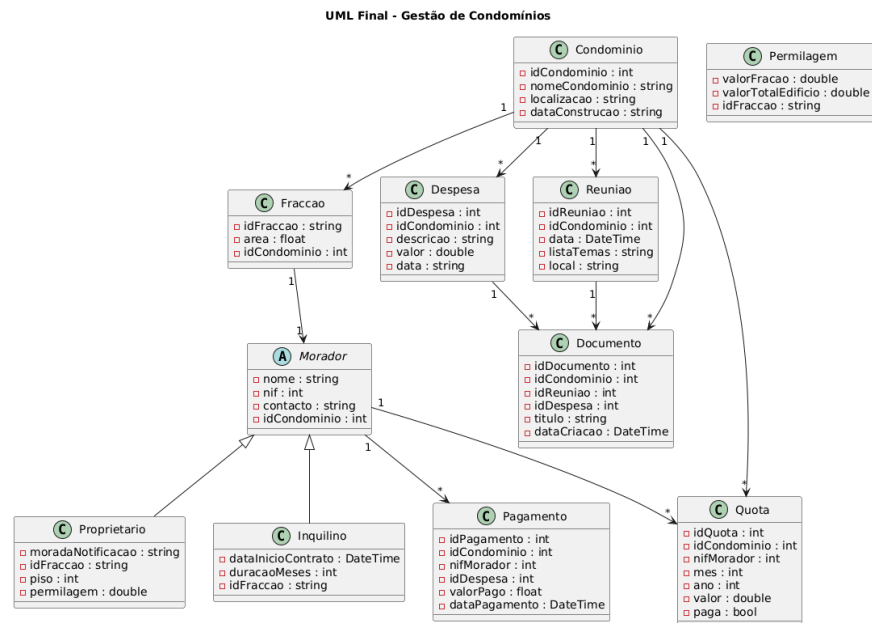


Figura 2 – Diagrama UML final

## 7. Conclusão

O presente trabalho permitiu desenvolver um sistema de **Gestão de Condomínios** recorrendo aos princípios fundamentais da **Programação Orientada a Objetos**, aplicados de forma progressiva ao longo das duas fases do projeto.

Na **Fase 1**, foi definido o modelo base do sistema, com as classes principais, relações e persistência em ficheiros.

Na **Fase 2**, o sistema foi evoluído através da introdução de **Regras de Negócio**, **testes unitários** e uma melhor separação de responsabilidades entre camadas.

A utilização de herança, encapsulamento e modularização contribuiu para um código mais organizado, coerente e de fácil manutenção. A implementação de testes unitários, ainda que limitada, permitiu validar o correto funcionamento de partes críticas do sistema.

De forma global, os objetivos propostos foram cumpridos, resultando num sistema funcional, extensível e adequado ao contexto académico da unidade curricular.



## 8 - Referências

**StackOverflow. (2025).** Website para ver excertos de código de outras pessoas. Disponível em:  
<https://stackoverflow.com/>

**W3Schools. (2025).** Website utilizado para estudos sobre programação no geral. Disponível em:  
<https://www.w3schools.com/cs/>

**GitHub. (2025).** Repositório com relatório de inspiração. Disponível em:  
[https://github.com/LFtech6/EDA\\_TP](https://github.com/LFtech6/EDA_TP)

**GitHub. (2025).** Repositório das aulas. Disponível em:  
<https://github.com/luferIPCA/LESI-POO-2025-2026>

**ChatGPT. (2025).**  
<https://chatgpt.com>