Apresentação do Sistema de Gestão de Biblioteca

1. Introdução:

Nesta seção, você introduz o sistema de gestão de biblioteca e os pressupostos que você considerou ao criar o diagrama UML.

Pressupostos considerados na elaboração do diagrama:

- O sistema de gestão de biblioteca deve cobrir as operações essenciais, como cadastro de livros, usuários e a gestão de empréstimos.
- O sistema deve permitir a consulta de disponibilidade de livros, empréstimos e devoluções, além de gerar relatórios sobre as transações realizadas.
- Foram modeladas entidades-chave como **Biblioteca**, **Livro**, **Utilizador** (separando em tipos como **Aluno** e **Professor**), e **Empréstimo**, considerando suas interações e comportamentos.
- Considerou-se que a biblioteca pode ter múltiplos livros, múltiplos utilizadores, e que os empréstimos e devoluções de livros seriam gerenciados diretamente por um sistema automatizado.

---

2. Desenvolvimento:

Nesta parte, você mostra o diagrama e explica cada uma das classes e suas relações, detalhando seus atributos e métodos.

Exibição do diagrama:

- Exiba o diagrama UML gerado pelo PlantUML.

Explicação detalhada do diagrama:

1. **Classe Biblioteca**:
    - Atributos:
        - `nome`: Nome da biblioteca.
        - `endereço`: Endereço da biblioteca.
    - Métodos:
        - `registrar_livro()`: Registra novos livros na biblioteca.
        - `registrar_utilizador()`: Registra novos utilizadores (alunos ou professores).

        - gerar_relatorio(): Gera relatórios de livros emprestados e devolvidos.
2. **Classe ItemBiblioteca (classe base abstrata)**:
    - Métodos:
        - consultar_disponibilidade(): Verifica se o item (livro) está disponível para empréstimo.
    - **Livro** herda de **ItemBiblioteca**.
3. **Classe Livro**:
    - Atributos:
        - título: O título do livro.
        - autor: O autor do livro.
        - ISBN: Número único de identificação do livro.
        - categoria: A categoria do livro (ex: ficção, ciência, etc.).
        - disponibilidade: Indica se o livro está disponível para empréstimo.
    - Métodos:
        - emprestar(): Realiza o empréstimo do livro.
        - devolver(): Registra a devolução do livro.
4. **Classe Utilizador (classe base para Aluno e Professor)**:
    - Atributos:
        - nome: Nome do utilizador.
        - endereço: Endereço do utilizador.
        - telefone: Número de telefone para contato.
    - Métodos:
        - registrar(): Registra o utilizador no sistema.
5. **Classe Aluno (herda de Utilizador)**:
    - Atributos:
        - matrícula: Número de matrícula do aluno.
6. **Classe Professor (herda de Utilizador)**:
    - Atributos:
        - departamento: Departamento a que o professor pertence.
7. **Classe Emprestimo**:
    - Atributos:
        - data_emprestimo: A data em que o livro foi emprestado.
        - data_devolucao: A data de devolução do livro.
    - Métodos:
        - realizar_emprestimo(): Registra o empréstimo de um livro.
        - registrar_devolucao(): Registra a devolução de um livro.
8. **Classe Relatório**:
    - Atributos:
        - data_geração: A data em que o relatório foi gerado.
    - Métodos:
        - gerar_relatorio(): Gera um relatório sobre os empréstimos e devoluções realizados.

Relações:

- **Biblioteca** está associada a **Livro** e **Utilizador** em uma relação de "0..*" para cada um, representando que uma biblioteca pode ter múltiplos livros e utilizadores.
- **Livro** tem uma relação de 1 para muitos com **Empréstimo**, pois cada livro pode ser emprestado várias vezes.
- **Utilizador** tem uma relação de "0..*" com **Empréstimo**, pois um utilizador pode fazer múltiplos empréstimos.
- **Livro** herda de **ItemBiblioteca**, que define o comportamento básico de um item da biblioteca.
- **Aluno** e **Professor** herdam de **Utilizador**, representando os diferentes tipos de utilizadores do sistema.

---

3. Conclusão:

Nesta seção, você discute os obstáculos encontrados durante o processo de modelação e como eles foram superados.

Obstáculos e como foram superados:

- **Dificuldade em identificar as classes principais**: No início, foi desafiador definir quais classes seriam essenciais para o sistema. Após revisar as funcionalidades e objetivos do sistema, ficou claro que as classes principais seriam aquelas relacionadas à gestão de livros e empréstimos.
- **Gerenciamento de heranças**: Outro obstáculo foi determinar como organizar as heranças entre classes como **Utilizador**, **Aluno** e **Professor**. A solução foi aplicar a herança para evitar duplicação de código, definindo **Aluno** e **Professor** como subclasses de **Utilizador**, já que compartilham a maioria dos atributos e métodos.
- **Relações complexas**: Definir as relações entre as classes (especialmente entre **Livro**, **Utilizador** e **Empréstimo**) exigiu atenção para garantir que o diagrama refletisse corretamente as interações no sistema. Superamos isso definindo claramente os multiplicadores e responsabilidades de cada entidade no sistema.