Relatório para a recuperação de POO

Aluno: Pedro Luis Olivo da Silva
Instituto Federal do Paraná - Campus Pinhais
Bacharelado em Ciência da Computação 2° Semestre

19/01/2025

1 - Explicando os conceitos de POO

1.1 - Herança

```java
public abstract class Usuario {

    protected int id;
    protected String nome;
    protected String dataNascimento;
    protected String email;
    protected String endereco;
    protected String telefone;
    protected List<LivroModel> listaDeLivros = new ArrayList<>();

}
```

Usuário é a classe pai de Aluno e professor.

```java
package models;

public class AlunoModel extends Usuario {

    public AlunoModel(int id, String nome, String dataNascimento, String email, String endereco, String telefone){
        this.id = id;
        this.nome = nome;
        this.dataNascimento = dataNascimento;
        this.email = email;
        this.endereco = endereco;
        this.telefone = telefone;
    }
```

```java
package models;

public class ProfessorModel extends Usuario {

    public ProfessorModel(int id, String nome, String dataNascimento, String email, String endereco, String telefone){
        this.id = id;
        this.nome = nome;
        this.dataNascimento = dataNascimento;
        this.email = email;
        this.endereco = endereco;
        this.telefone = telefone;
    }
```

Nesse caso utilizo usuário para ser a classe pai de aluno e professor que terão os mesmos atributos, no código em si os dois são iguais, mas poderia ser implementado atributos diferentes para cada um.

1.2 - Polimorfismo

```java
public interface Icrud {

    public void cadastrar();
    public void listar();
    public void editar();
    public void remover();
}
```

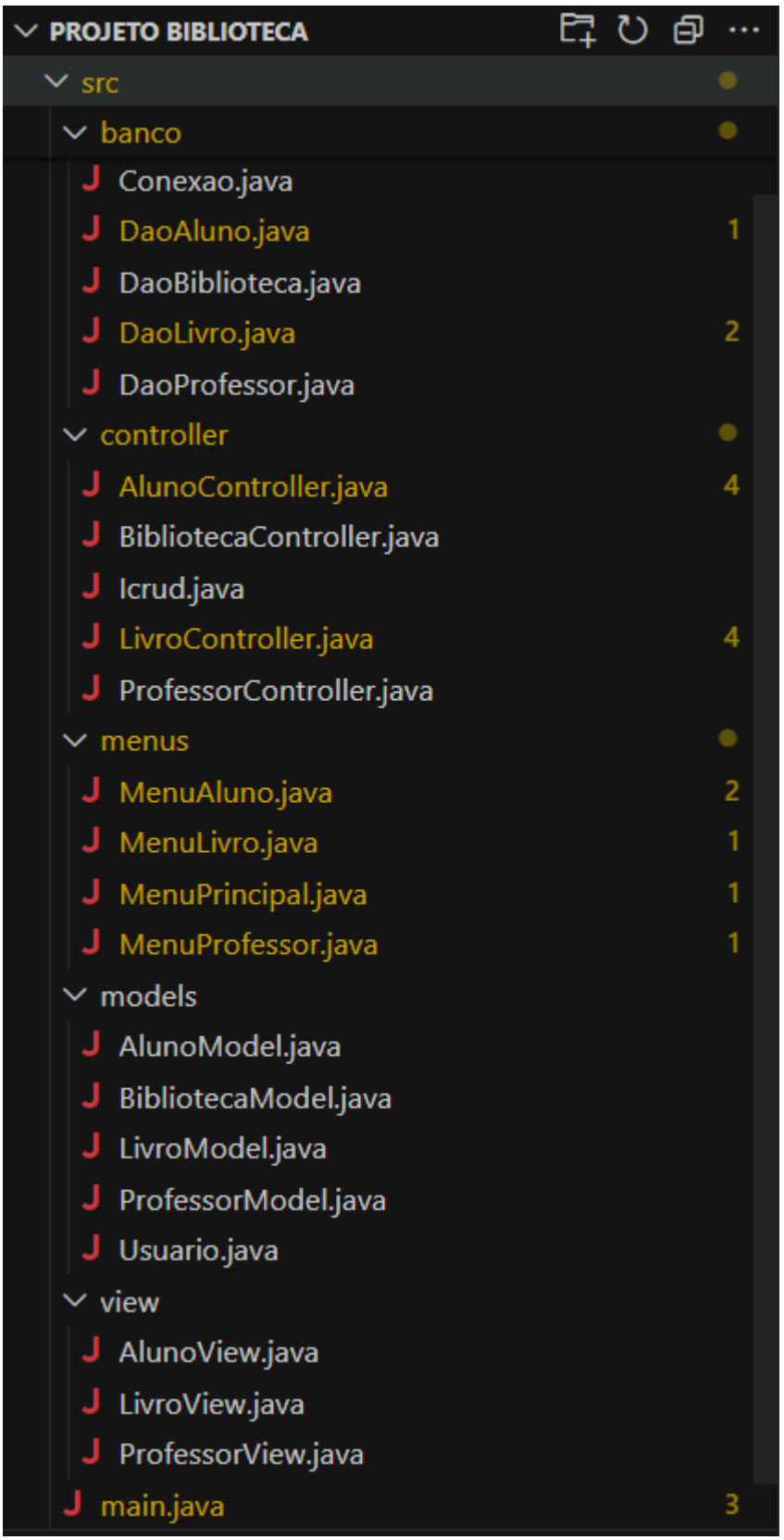
```java
@Override
public void cadastrar(){
    LivroModel livroModel = new LivroModel();
    view.infoLivro(livroModel);
    daoLivro.cadastroLivro(livroModel);
    bibliotecaController.salvarLivros(livroModel);

}
```

```java
@Override
public void cadastrar(){
    view.infosAluno(alunoModel);
    daoAluno.cadastroAluno(alunoModel);
}
```

O polimorfismo está presente nas variáveis de mesmo nome que podem ter funções diferentes. Na primeira imagem está representada a interface do código que é usada para aplicar diferentes funções.

1.3 - Abstração



As classes estão separadas em pastas onde cada uma tem sua função, a pasta banco tem a parte da conexão com o banco de dados e o Dao(Data Access Object) um padrão de

projeto que em muitas das suas funções uma é a de executar os comandos SQL. O controller tem a classe Icrud que é a interface utilizada nos demais controllers que aplicam os métodos do código. Os menus estão ligados ao controller e servem apenas para interagir com o usuário. As models guardam os atributos das classes principais. A view também serve para interagir com o usuário. A main que não está dentro de nenhuma classe roda o código.

1.4 - Encapsulamento

```
public class LivroModel {

    protected int id;
    protected String nome;
    protected String autor;
```

O encapsulamento existe para proteger os atributos e métodos de uma classe(getters e setters por exemplo). Eles são representados pelo protected antes de declarar uma variável.

2 - Padrões utilizados

O único padrão de projeto utilizado é o DAO(Data Access Object) que faz a função de encapsular a parte do banco de dados do código.

3 - Dificuldades na implementação

Não foi possível implementar o sistema de empréstimo e devolução, as classes BibliotecaModel, BibliotecaController e DaoBiblioteca foram a tentativa da implementação.

A BibliotecaModel guardaria três listas de livros, livrosCadastrados que guardaria os livros para não alterar seus ids nas trocas de listas, livrosDisponiveis que guardaria os livros disponiveis para o empréstimo e livrosOcupados que guardaria os livros ocupados.

A BibliotecaController faria as operações de trocas de livros entre as listas. O DaoBiblioteca jogaria as informações para o banco de dados.

No banco seriam criadas tabelas para representar as listas, porém elas teriam que ter uma conexão, algo que não sei como implementar.

Fora a parte de empréstimo e devolução o desenvolvimento do resto do código foi mais tranquilo, dando ênfase na dificuldade da conexão com banco que eu nunca tinha mexido.