Desenvolvimento Full Stack

Felipe Marques de Almeida - 202208291929

Polo Rodovia Br 407 - Juazeiro - Ba
RPG0016 - BackEnd sem banco não tem – Turma 22.3 – 3º Semestre
GitHub: Kifflom2108/Missao-Pratica-Nivel-3-Mundo-3 (github.com)

Objetivo da Prática

Implementar persistência com base no middleware JDBC.

Utilizar o padrão DAO (Data Access Object) no manuseio de dados.

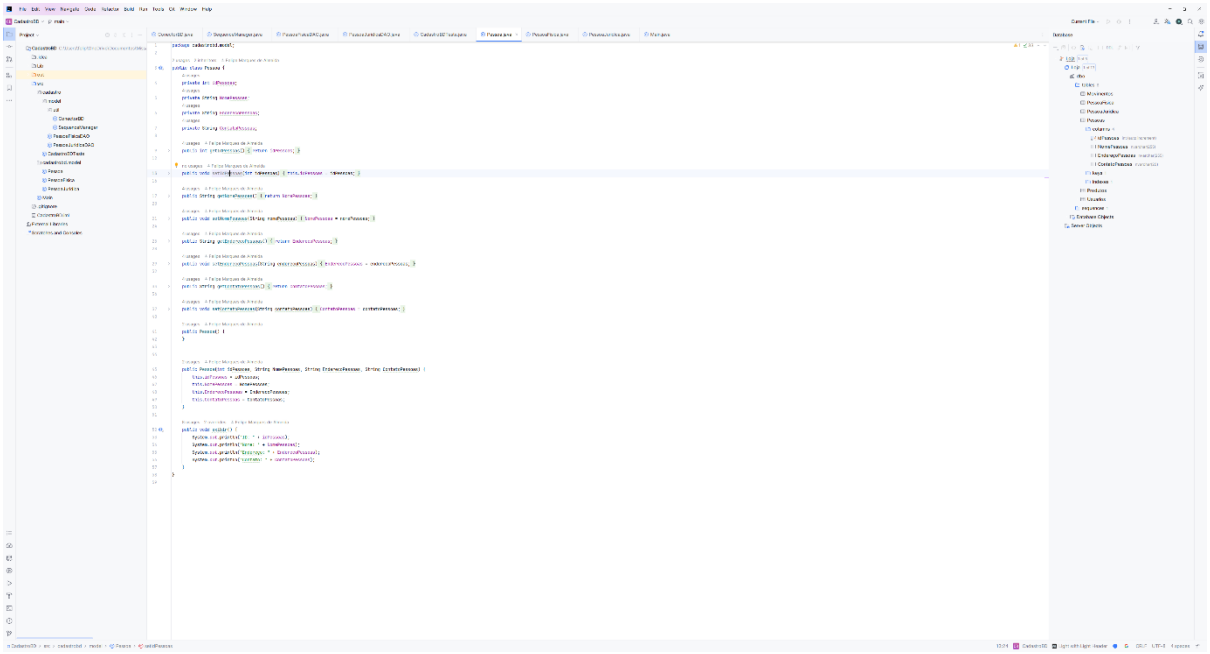
Implementar o mapeamento objeto-relacional em sistemas Java.

Criar sistemas cadastrais com persistência em banco relacional.

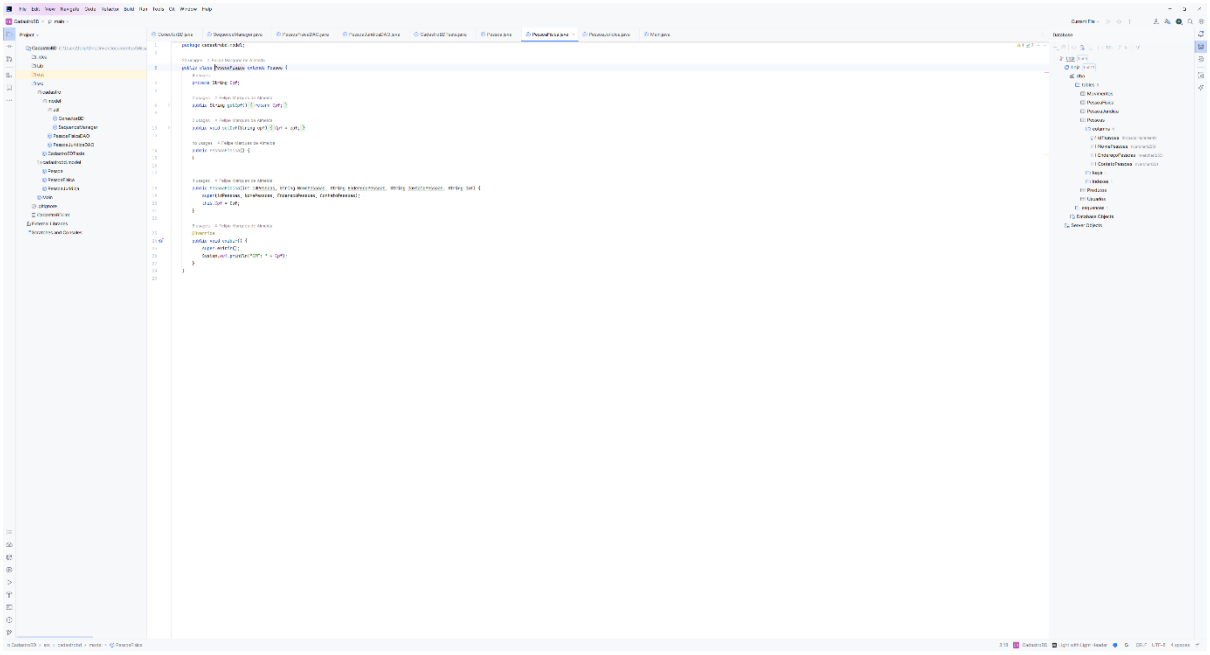
No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

Códigos Desenvolvidos:

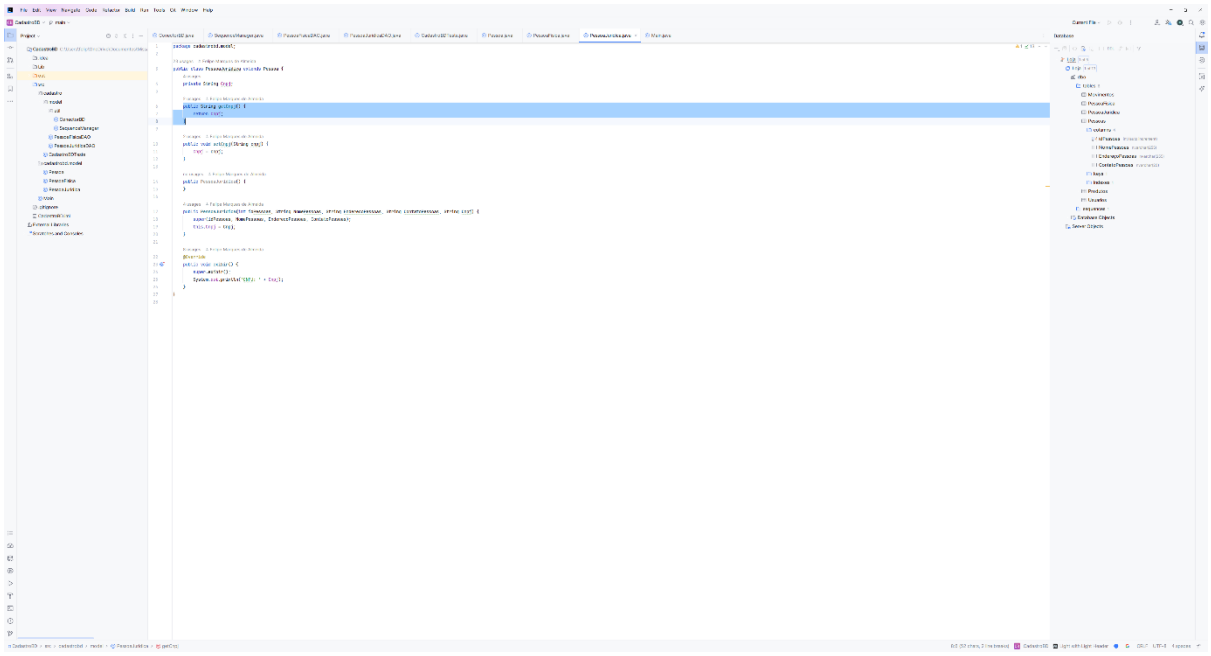
Pessoa :



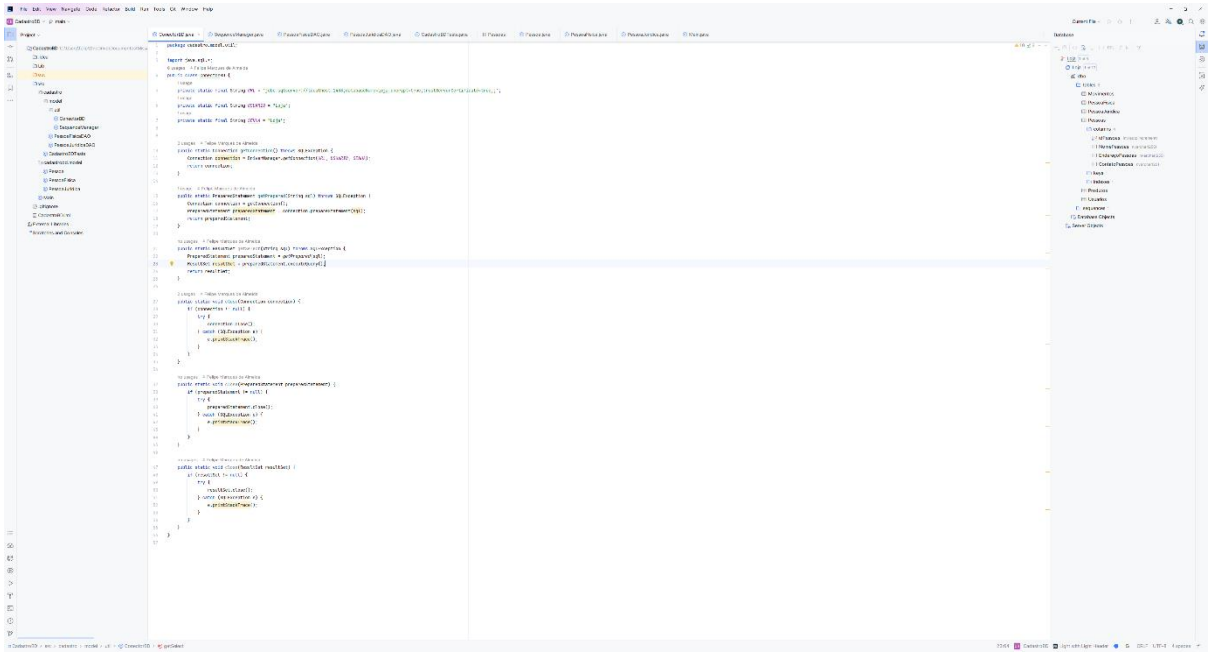
PessoaFisica :



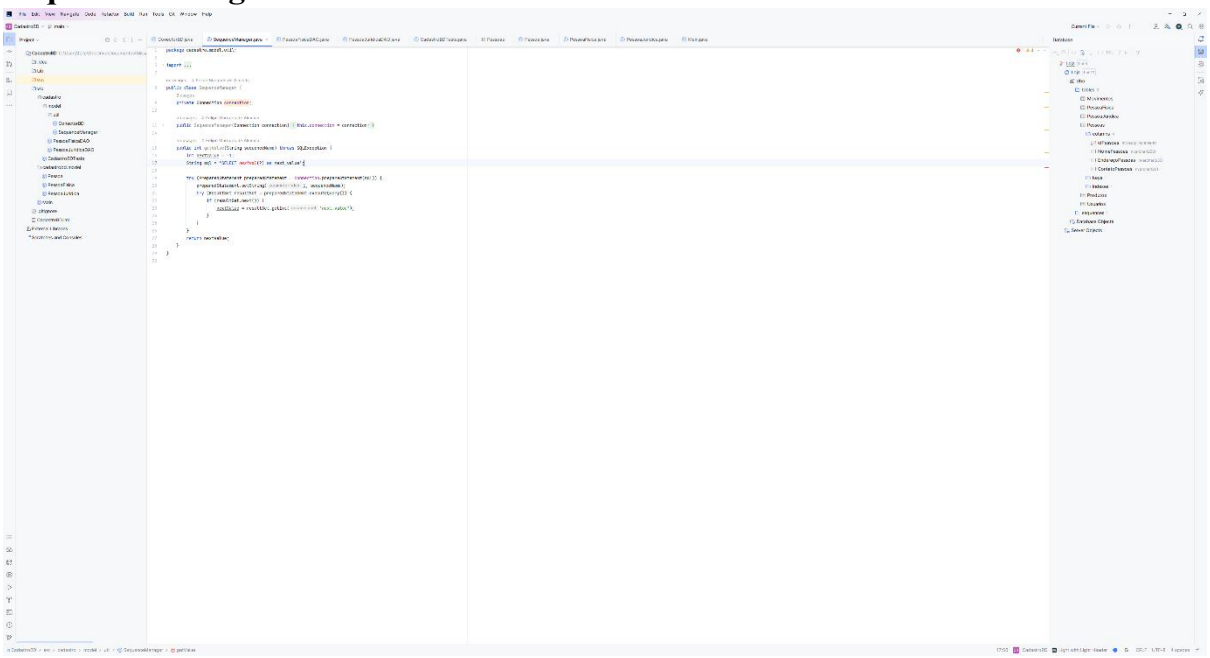
PessoaJuridica :



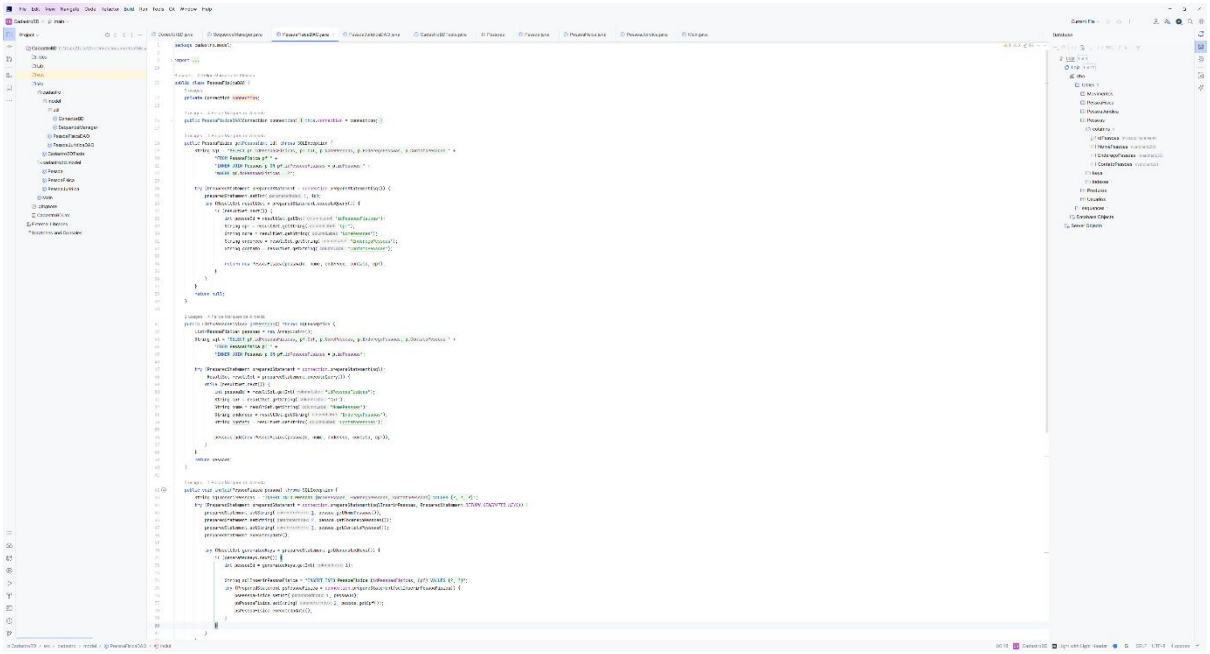
ConectorBD :

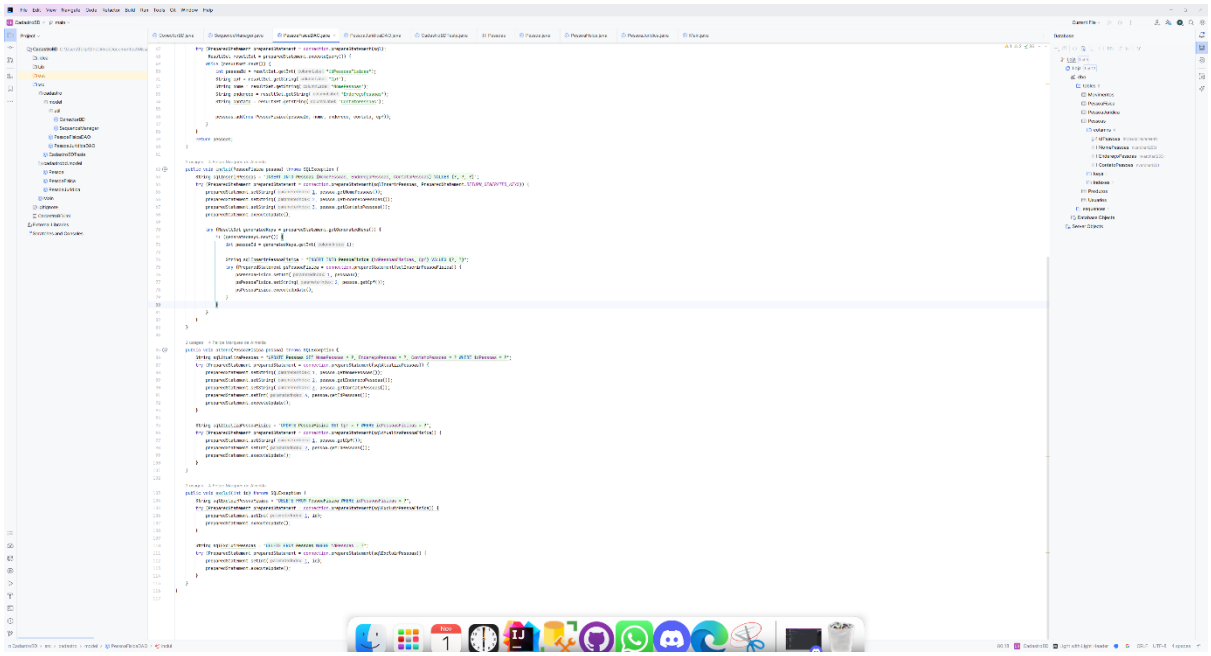


SequenceManager :

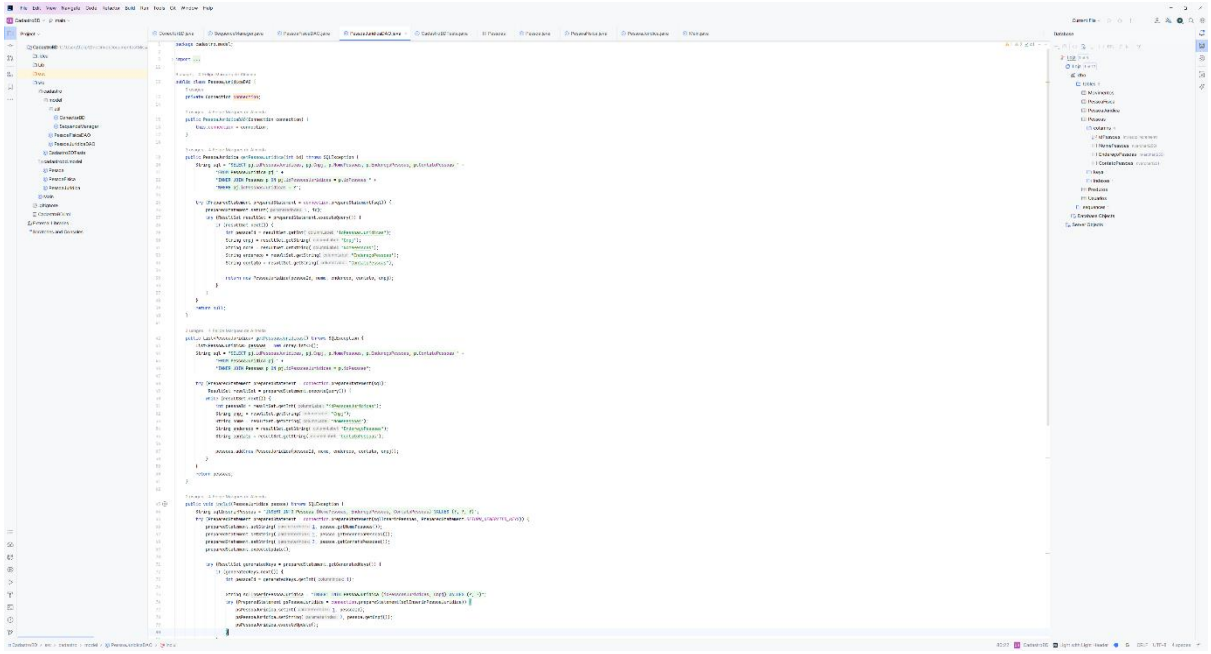


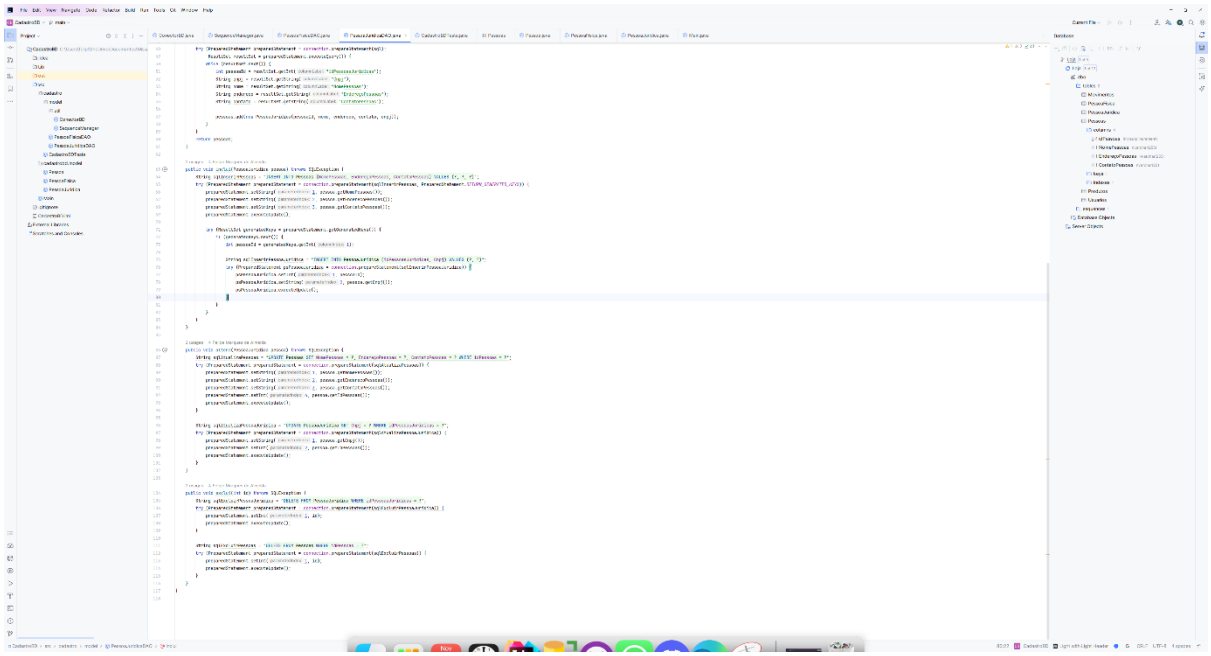
PessoaFisicaDAO :



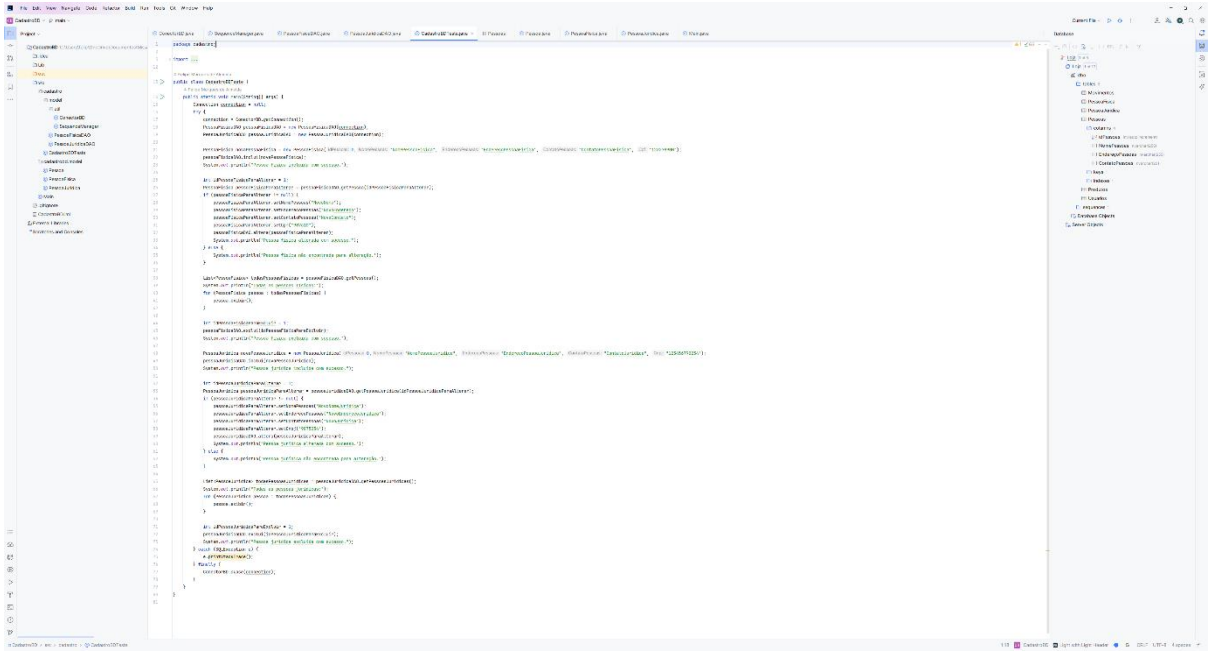


PessoaJuridicaDAO :

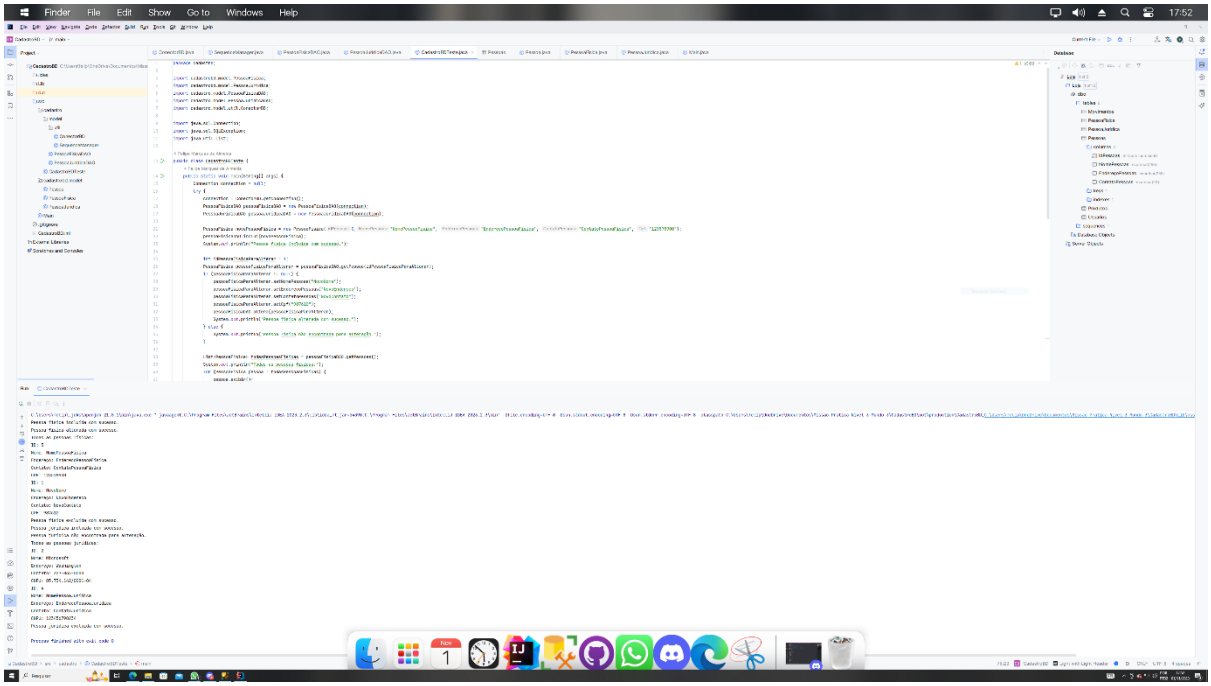




CadastroBDTeste :



Resultado do CadastroBDTeste :



A Importância dos Componentes de Middleware, como o JDBC

No mundo da programação e desenvolvimento de software, os componentes de middleware desempenham um papel crucial na interação entre aplicativos e sistemas de banco de dados. Um exemplo notável de middleware é o JDBC (Java Database Connectivity), que é particularmente relevante para aplicativos Java. O JDBC atua como uma ponte essencial entre um programa Java e um banco de dados, permitindo que os desenvolvedores acessem e manipulem os dados de forma eficiente e segura. Isso é de suma importância, pois simplifica a conexão com bancos de dados, garantindo uma comunicação eficaz e segura entre aplicativos e sistemas de gerenciamento de banco de dados.

Diferença entre Statement e PreparedStatement na Manipulação de Dados

Ao trabalhar com JDBC ou outros mecanismos de acesso a bancos de dados, é importante entender a diferença entre Statement e PreparedStatement. O Statement é uma interface que permite executar consultas SQL simples, mas muitas vezes vulneráveis a injeções de SQL. Por outro lado, o PreparedStatement é uma subclasse do Statement que permite a criação de consultas parametrizadas, tornando-as mais seguras e eficientes. A principal diferença está na compilação e reutilização de consultas preparadas, o que reduz a sobrecarga e melhora o desempenho, além de proteger contra ataques de injeção SQL.

O Padrão DAO e sua Contribuição para a Manutenibilidade do Software

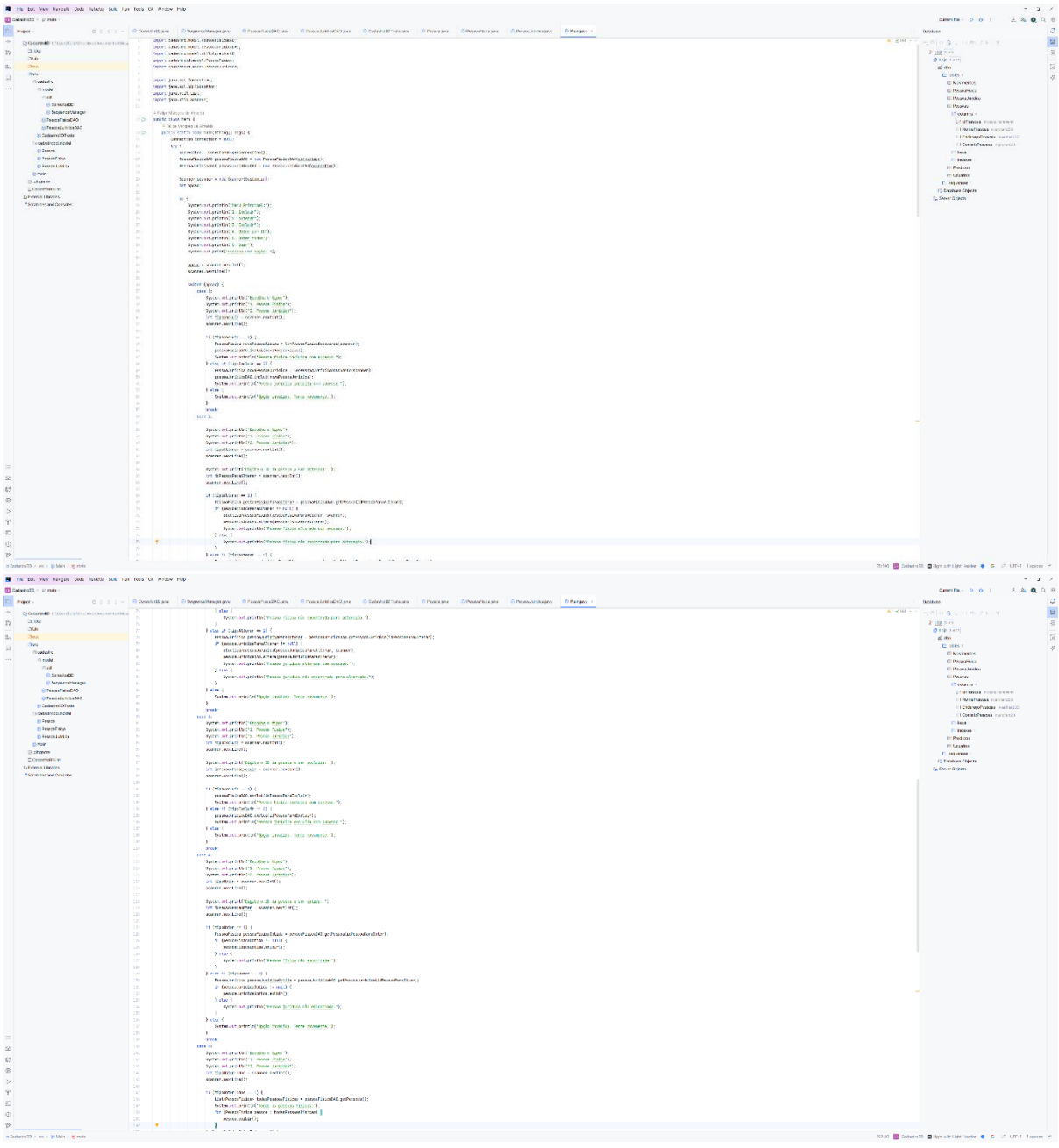
O padrão DAO (Data Access Object) desempenha um papel crucial na melhoria da manutenibilidade do software. O DAO separa a lógica de acesso a dados da lógica de negócios, proporcionando uma abstração que facilita a manutenção e a evolução do código. Ao encapsular a complexidade do acesso a bancos de dados em classes DAO dedicadas, as alterações na estrutura do banco de dados têm menos impacto nas partes do código que dependem dessas classes. Isso permite uma manutenção mais eficiente e escalável do software, tornando-o mais flexível para adaptações futuras.

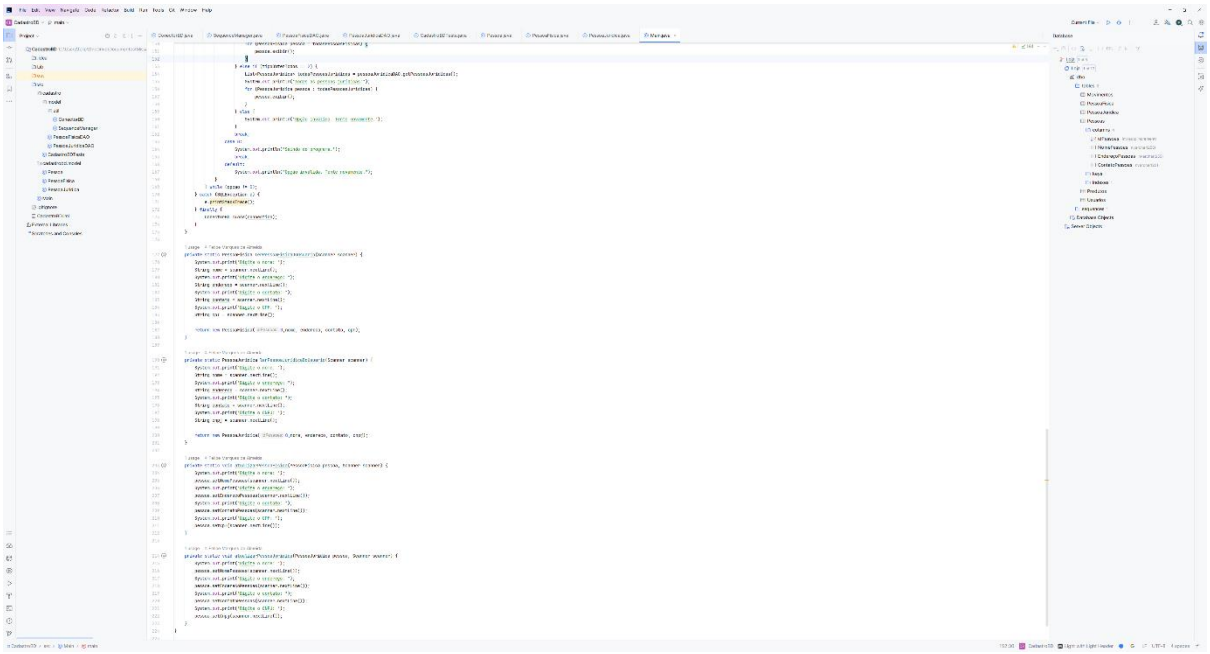
Reflexo da Herança em um Modelo Estritamente Relacional no Banco de Dados

Quando lidamos com um modelo estritamente relacional em um banco de dados, a herança é frequentemente refletida por meio de técnicas de modelagem específicas. A herança em bancos de dados relacionais é implementada por meio de tabelas e relacionamentos que representam as hierarquias de classes. As tabelas filhas compartilham chaves primárias com a tabela pai, permitindo a busca e consulta de dados de acordo com a hierarquia. Isso facilita a representação de objetos de classes derivadas em um banco de dados relacional, mas requer uma modelagem cuidadosa para garantir a integridade e a consistência dos dados.

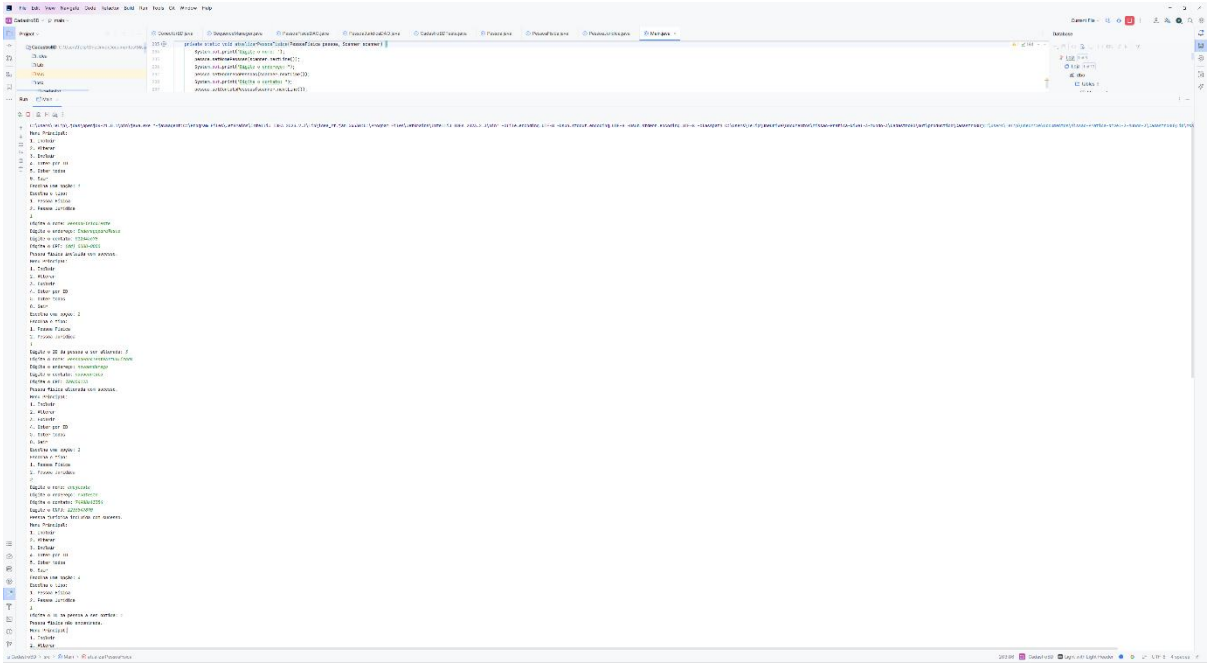
Em resumo, os componentes de middleware, como o JDBC, são fundamentais para a comunicação eficaz entre aplicativos e bancos de dados. A escolha entre Statement e PreparedStatement afeta a segurança e o desempenho na manipulação de dados. O padrão DAO melhora a manutenibilidade do software, enquanto a reflexão da herança em um modelo estritamente relacional no banco de dados é um desafio de modelagem essencial para representar hierarquias de classes de forma consistente e eficaz. Considerar esses aspectos é fundamental para o sucesso no desenvolvimento de aplicativos e sistemas de gerenciamento de banco de dados.

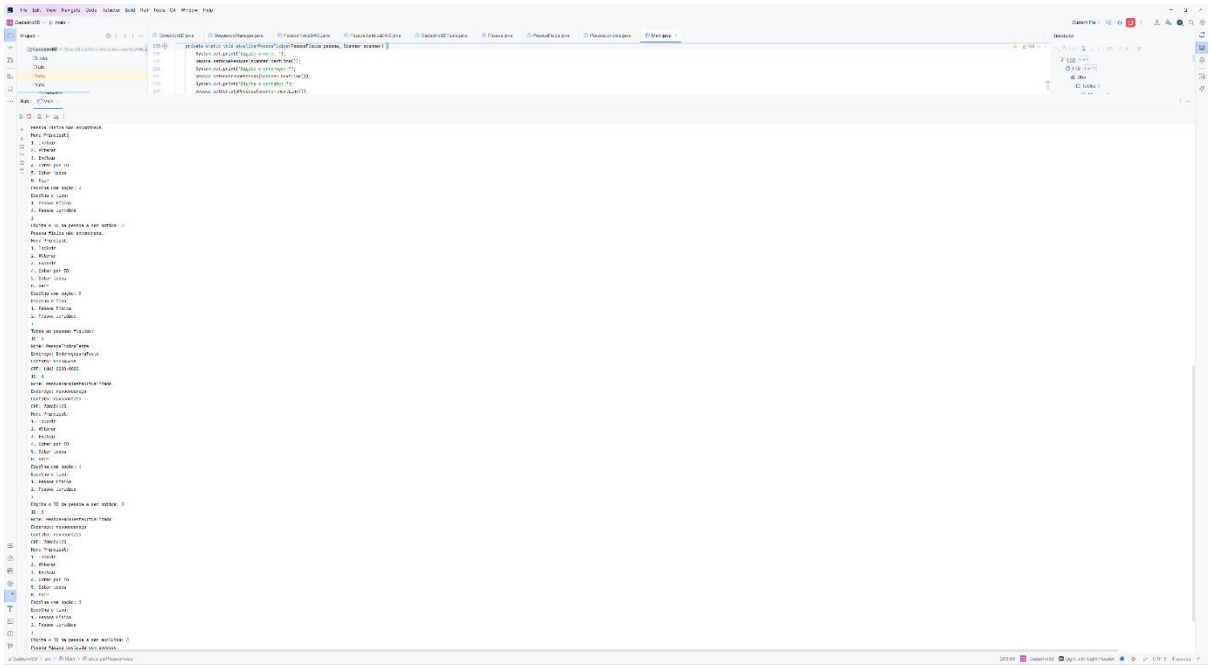
Main :





Resultado Main :





Diferenças entre Persistência em Arquivo e Persistência em Banco de Dados

A persistência em arquivo e a persistência em banco de dados são duas abordagens diferentes para armazenar e recuperar dados em um sistema de software. A principal diferença entre elas reside na forma como os dados são armazenados e acessados:

Persistência em Arquivo: Nesse método, os dados são armazenados em arquivos no sistema de arquivos do computador. Geralmente, são usados formatos como texto, JSON ou binário. A persistência em arquivo é adequada para pequenas quantidades de dados e é relativamente simples de implementar. No entanto, pode ser menos eficiente na busca e manipulação de dados em comparação com bancos de dados. Além disso, a escalabilidade e a concorrência podem ser desafios.

Persistência em Banco de Dados: Nesse método, os dados são armazenados em um sistema de gerenciamento de banco de dados (SGBD), como o MySQL, PostgreSQL ou SQL Server. Os bancos de dados oferecem recursos avançados, como indexação, consulta complexa e suporte a transações, o que torna a manipulação de dados mais eficiente e segura. A persistência em banco de dados é apropriada para sistemas que precisam lidar com grandes volumes de dados, garantir a consistência e a integridade dos dados e suportar operações concorrentes.

Uso de Operador Lambda na Impressão de Valores em Versões Mais Recentes do Java

O uso de operadores lambda nas versões mais recentes do Java, a partir do Java 8, simplificou a impressão de valores contidos em entidades (objetos) ao introduzir uma maneira mais concisa de expressar funções anônimas. Com operadores lambda, você pode definir comportamentos de forma mais compacta, o que é especialmente útil ao lidar com listas, coleções ou streams de dados.

Métodos Acionados Diretamente pelo Método Main e a Necessidade de Serem Marcados como Static

Métodos acionados diretamente pelo método main em uma classe Java precisam ser marcados como static devido à forma como o método main é definido. O método main é o ponto de entrada de um programa Java e é invocado pelo ambiente de execução sem a necessidade de criar uma instância da classe que o contém. Para ser chamado diretamente, um método também deve ser static, o que significa que ele pertence à classe e não a instâncias individuais da classe.

A marcação de um método como static permite que ele seja chamado sem a necessidade de criar um objeto da classe que o contém. É uma característica essencial para métodos que desempenham funções globais, como o método main, que inicia a execução de um programa Java. Portanto, a marcação de métodos acionados diretamente pelo main como static é uma convenção que reflete a natureza do método main como um método de classe e não de instância.