UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS EXATDAS E NATURAIS
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA E TECNOLOGIA
PROJETO FINAL (2024)

DESENVOLVIMENTO DE SISTEMA DE CADASTRO E LOGIN PARA O NÚCLEO DE PESQUISA DE BAIXO CARBONO

Rusemildo Alves
Gabriel Almeida
Ian Kairo
Sérgio Carvalho

RESUMO
Este artigo descreve o desenvolvimento de um sistema de cadastro e login voltado para os usuários do Núcleo de Pesquisa de Baixo Carbono (NPCO2) da Universidade Federal Rural do Semi-Árido (UFERSA). O objetivo é registrar os dados dos usuários e as análises realizadas, garantindo maior controle e eficiência no processamento das informações relacionadas às pesquisas de baixo impacto ambiental, como biochar e energias renováveis. O sistema foi implementado em Java, utilizando a bibliotecas para a criação da interface gráfica. Resultados preliminares indicam que o sistema atende aos requisitos propostos, proporcionando uma interface intuitiva e eficaz para os usuários.

Palavras-chave: POO, Java, Registro de Usuário, Interface Grafica

ABSTRACT
This article describes the development of a registration and login system aimed at users of the Low Carbon Research Center (NPCO2) of the Federal Rural University of the Semi-Arid (UFERSA). The objective is to record user data and the analyzes carried out, ensuring greater control and efficiency in the processing of information related to research with low environmental impact, such as biochar and renewable energies. The system was implemented in Java, using the Swing library for the creation of the graphical interface. Preliminary results indicate that the system meets the proposed requirements, providing an intuitive and effective interface for users.

Key words: OOP, Java, User registration, Graphical user interface.

1 INTRODUÇÃO

O Núcleo de Pesquisa de Baixo Carbono (NPCO2) da UFERSA desenvolve soluções sustentáveis que contribuem para a redução das emissões de gases de efeito estufa, em especial o dióxido de carbono (CO_2). Para garantir o sucesso das atividades, é essencial um sistema eficiente de cadastro e controle das análises realizadas pelos pesquisadores. Desta forma, o artigo descreve desenvolvimento de um sistema para o gerenciamento dessas informações, utilizando os conhecimentos em programação orientada a objetos em Java para criar um código com o objetivo que os usuários do Laboratório NPCO2 consigam fazer um cadastro e para registrar análises feitas por eles.

O intuito deste trabalho é criar um sistema que atenda às necessidades específicas do NPCO2 para as análises realizadas. Para garantir um controle eficaz e otimizado dessas análises, é essencial a construção de um banco de dados. Esse banco permitirá organizar as informações de forma estruturada, facilitando o armazenamento, a consulta, a atualização e o monitoramento dos

dados, possibilitando um fluxo de trabalho otimizado e seguro no tratamento das informações. A escolha do Java como linguagem para a implementação desse projeto se deu devido a ser uma linguagem estática e orientada a objetos, o que garante maior controle e menos erros no desenvolvimento. Além disso, sua gestão de memória é automática, evitando problemas comuns em outras linguagens.

2 DESENVOLVIMENTO

Este tópico do artigo abordara no núcleo do trabalho a ser apresentado. Onde acontecera a apresentação da fundamentação teórica necessária para o ingresso ao tema junto com os objetivos e a metodologia adotada no trabalho.

2.1 Fundamentação teórica

Este estudo utiliza a fundamentação teórica da POO, os benefícios da linguagem Java e a robustez dos bancos de dados relacionais para desenvolver um sistema de cadastro e registro de análises. Ao integrar esses elementos, buscando criar uma solução eficiente e escalável para o Laboratório Núcleo de Pesquisa de Baixo Carbono (NPCO2)

2.1.1 Programação Orientada a Objetos (POO)

Nos anos 90, muitas empresas já haviam informatizado boa parte de seus sistemas e a internet se tornou uma ferramenta comum para comunicação e busca de informações. Com isso, surgiu a demanda por softwares mais atraentes, dinâmicos e que oferecessem alta capacidade de troca de dados. Esses novos softwares precisavam interagir melhor com os usuários, apresentar interfaces gráficas e serem adaptáveis a mudanças rápidas de hardware, além de facilitar a comunicação entre diferentes sistemas e serem portáteis para várias plataformas e sistemas operacionais [3].

Pesquisas mostraram que a reutilização de código era fundamental para aumentar a produtividade e melhorar a qualidade dos softwares. Reutilizar significa aproveitar recursos já desenvolvidos, o que pode acelerar o processo de desenvolvimento. As técnicas de programação estruturada se mostraram insuficientes para atender a essas novas demandas, levando os programadores a adotarem a Programação Orientada a Objetos como uma abordagem mais eficaz [3].

A Programação Orientada a Objetos (POO) é uma abordagem que vê os sistemas como coleções de objetos interativos. Essa técnica melhora a reutilização e a extensibilidade dos softwares. A POO é baseada em um modelo de objetos que inclui princípios como abstração, hierarquização, encapsulamento, classificação, modularização, relacionamento, simultaneidade e persistência [3].

Um dos principais diferenciais da programação orientada a objetos em comparação com outros paradigmas que também permitem a definição de estruturas e operações é o conceito de herança. Esse mecanismo possibilita a extensão fácil de definições já existentes. Além da herança, é essencial destacar a relevância do polimorfismo, que permite a seleção dinâmica de funcionalidades que um programa utilizará durante sua execução [4].

2.1.2 Java como Linguagem de Programação

Java é uma linguagem de programação versátil, ideal para desenvolver aplicações em redes, como a Internet, redes privadas e programas independentes. Criada na primeira metade dos anos 90 nos laboratórios da Sun Microsystems, seu objetivo era ser mais simples e eficiente que linguagens anteriores, focando inicialmente em software para produtos eletrônicos, como micro-ondas. Um requisito importante era ter um código compacto e uma arquitetura neutra [6].

Apesar de ter atendido a esses objetivos, Java não teve sucesso comercial imediato. Contudo, com o crescimento da Internet, a Sun Microsystems viu uma oportunidade para utilizar a linguagem nesse contexto. Adaptaram Java para que funcionasse em microcomputadores conectados à rede, permitindo a criação de applets, que são programas pequenos que interagem com navegadores.

Atualmente, Java é fundamental em muitos avanços tecnológicos, como: Acesso remoto a bancos de dados, Bancos de dados distribuídos, Comércio eletrônico na web, CAD em rede, Interatividade em páginas web, Ambientes de Realidade Virtual distribuídos, Gerenciamento de documentos, Integração de dados e visualizações, Computadores em rede, Ensino à distância, Jogos e entretenimento. Esses aspectos ressaltam a importância e a adaptabilidade da linguagem Java no cenário atual da tecnologia [6].

Java é uma linguagem de programação que se destaca pela sua simplicidade e facilidade de aprendizado, apresentando um número reduzido de construções. Essa característica ajuda a diminuir erros comuns, como aqueles relacionados a ponteiros e ao gerenciamento de memória. Além disso, Java inclui um extenso conjunto de bibliotecas que oferecem funcionalidades básicas, como acesso à rede e criação de interfaces gráficas. Baseada na Orientação a Objetos, Java permite agrupar dados (variáveis) e métodos em blocos de software, o que facilita a modularização das aplicações, o reaproveitamento de código e a manutenção de sistemas já desenvolvidos [6].

As linguagens de programação podem ser compiladas ou interpretadas. Nas linguagens compiladas, um programa traduz o código-fonte em código executável, resultando em aplicações de alta performance, mas que só funcionam em arquiteturas específicas. Em contraste, as linguagens interpretadas rodam apenas em código fonte, utilizando um interpretador que executa os comandos em tempo real, permitindo maior flexibilidade em diferentes plataformas e facilitando a depuração.

Java é um exemplo de linguagem que combina compilação e interpretação. Após escrever um programa em Java, ele é compilado em arquivos de classe, que contêm bytecodes, não executáveis diretamente pelos processadores [6]. Esses bytecodes podem ser executados em qualquer sistema por meio de um ambiente de execução Java, permitindo que o código seja escrito e compilado apenas uma vez.

Java também se destaca por reduzir erros de programação, com um processo de compilação que detecta problemas antes da execução. O tratamento de exceções ajuda a manter a consistência do programa em caso de falhas [6]. Além disso, a linguagem implementa um esquema de segurança rigoroso, especialmente para códigos provenientes de redes inseguras, garantindo que o interpretador verifique e controle o acesso à memória e ao sistema de arquivos, respeitando permissões específicas de acordo com a origem das aplicações [6].

O Java API (Interface de Programação de Aplicações) é um conjunto de bibliotecas inclusas no JDK. Essas bibliotecas reúnem diversas classes organizadas em pacotes, cada um com funcionalidades fundamentais para

diferentes áreas da programação em Java. Pacotes como java.awt, java.io, java.sql [6].

O pacote java.awt inclui classes voltadas para a interface gráfica, como botões, caixas de texto e menus. Também abrange classes para processamento de imagens e tratamento de eventos gerados pela interface, como cliques de mouse e seleções de menu, disponíveis no subpacote java.awt.event. O pacote javax.swing é uma parte da Java API que fornece um conjunto de componentes gráficos para a construção de interfaces de usuário (UI) em aplicações Java. Ele é uma extensão do pacote java.awt e permite a criação de interfaces mais modernas e ricas, utilizando componentes leves que não dependem diretamente da plataforma subjacente.

Já o pacote java.io fornece classes para a entrada e saída de dados em diversas formas. Ele abstrai do usuário as particularidades do sistema de arquivos da plataforma em que o programa está sendo executado.

O pacote java.sql é parte da Java API e fornece classes e interfaces para a manipulação de dados em bancos de dados relacionais. Ele permite a conexão com diferentes sistemas de gerenciamento de banco de dados (SGBDs), além de possibilitar operações como consulta, inserção, atualização e exclusão de dados.

2.1.3 Banco de Dados

Um banco de dados pode ser descrito como uma coleção estruturada de informações, organizada para facilitar a busca e recuperação por um computador. Ele é mais do que uma simples reunião de itens. Existem vários tipos de bancos de dados, como hierárquicos, em rede, relacionais e orientados a objetos, cada um utilizando diferentes modelos de organização dos dados [5]. Por exemplo, os bancos de dados hierárquicos organizam as entradas em uma estrutura em árvore. Por sua vez, os bancos de dados orientados a objetos armazenam estruturas de dados complexas, chamadas de "objetos", que são dispostas em classes hierárquicas, podendo herdar características de classes superiores [5].

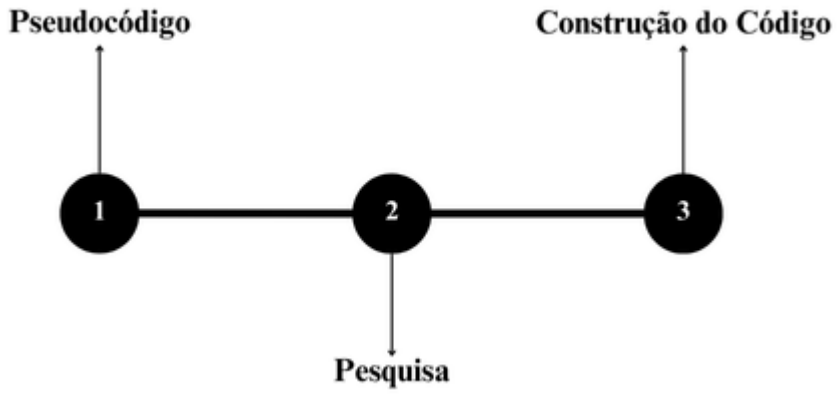
Os bancos de dados são fundamentais para o armazenamento e recuperação de dados em sistemas de informação. A integração de Java com bancos de dados relacionais, como o MySQL, é amplamente documentada e suportada por diversas bibliotecas e frameworks, como o JDBC (Java Database Connectivity).

2.2. Objetivos

O objetivo deste artigo é desenvolver um sistema eficiente de cadastro para usuários do Laboratório Núcleo de Pesquisa de Baixo Carbono (NPCO2) da UFERSA, bem como um sistema de registro das análises realizadas pelos usuários. O desenvolvimento será realizado utilizando a Programação Orientada a Objetos (POO) em Java, aliado a um banco de dados relacional para armazenamento das informações. Este sistema visa garantir um controle eficaz e otimizado das atividades do NPCO2, proporcionando um fluxo de trabalho organizado e seguro para a gestão das informações, alinhado com as melhores práticas de desenvolvimento de software.

2.3. Metodologia

Com base nos conceitos apresentados, foi estabelecido etapas para realização do trabalho.



Fluxograma 1: Etapas de Desenvolvimento. Fonte: Autoria Própria.

2.3.1. Pseudocódigo

Com base na necessidade do grupo NPCO2, e após estabelecer o seguinte projeto, desenvolver um cadastro para usuários do laboratório e um registro de análise feita pelo usuário. Foi desenvolvido o pseudocódigo a seguir.

Criar uma Interface do Projeto:

- Criar uma janela (frame), com duas opções Login e Cadastro, a partir da seleção das opções vai seguir dois caminhos de códigos diferentes.
- Criar no MYSQL, um database e 2 tabelas para armazenamentos de dados. Uma tabela de cadastro de usuário e outra de registro de análise.
- Criar uma classe database, com a conexão do SQL e Java.

Cadastro de cada Usuário do Laboratório:

- Criar uma Classe chamada Usuário. Onde recebe nome, senha, data de nascimento e projeto que faz parte.
- Criar uma classe chamada cadastro de usuário, dentro da classe criar uma interface para realização do cadastro e integração com banco de dados.
- Devolve um ID de registro, para usuário utilizar como login junto com a senha.

Registro de Análise:

- Criação de uma Classe Login, para fazer o reconhecimento do Usuário e passar para próxima etapa.
- Criar uma Classe chamada Registro Análise. Onde recebe descrição da análise, recebe um arquivo em PDF e armazena na tabela de registro de análise.
- Devolve um número de lote do registro.

2.3.2. Pesquisa

A etapa 2 consistiu em realizar uma pesquisa para entender os conceitos que seriam aplicados na implementação do código, especialmente em relação à linguagem Structured Query Language (SQL), que, apesar de não ser abordada na disciplina, é essencial para o desenvolvimento do projeto. Após compreender seu funcionamento e suas configurações, foi necessário investigar como integrá-la com o Java e a forma de implementação conjunta.
Além disso, revisamos alguns conceitos relevantes ao conteúdo da disciplina, com o objetivo de evitar retrabalho. A maior parte da pesquisa concentrou-se em identificar bibliotecas e pacotes Java adequados tanto para a criação da interface gráfica quanto para a conexão com SQL. Essa etapa foi crucial para garantir o sucesso na implementação do projeto.

2.3.3. Construção do Código

A implementação foi conduzida com base no método de tentativa e erro, utilizando pseudocódigo como guia. A criação da classe Usuário, o uso de encapsulamento e a implementação dos métodos foram relativamente simples. No entanto, grande parte do desenvolvimento envolveu conceitos e técnicas para os quais não estávamos completamente preparados. O uso de ferramentas externas, como pesquisas e inteligência artificial, foi fundamental para o progresso, especialmente na integração com SQL. Essas ferramentas auxiliaram significativamente na compreensão e aplicação correta das consultas e operações no banco de dados. A implementação seguiu os passos a seguir.

1. Criação da Interface do Projeto (App): A primeira parte consistiu em desenvolver uma janela principal (frame) com duas opções: "Login" e "Cadastro". Dependendo da escolha do usuário, o código seguiria caminhos diferentes. Utilizando componentes gráficos, como botões e eventos de clique. A interface foi projetada para ser intuitiva, direcionando o usuário para a função adequada.

2. Banco de Dados no MySQL: Foi criado um banco de dados no MySQL, que contém duas tabelas: uma para o cadastro de usuários e outra para o registro de análises. A tabela de usuários armazena informações como nome, senha, data de nascimento e o projeto ao qual o usuário pertence. A segunda tabela é destinada ao armazenamento dos registros de análise, incluindo descrições e arquivos em formato PDF.

3. Classe Database (Conexão SQL e Java): Desenvolvemos uma classe chamada Database, que é responsável por estabelecer a conexão entre o Java e o MySQL. Utilizamos a biblioteca JDBC para realizar essa integração.

4. Cadastro de Usuário: Foi criada a classe Usuário, que contém os atributos essenciais: nome, senha, data de nascimento e o projeto ao qual o usuário pertence. Essa classe serve de modelo para armazenar as informações de cada novo usuário. A seguir, criamos uma classe CadastroUsuario, que contém a interface gráfica para o cadastro. Nessa interface, o usuário pode inserir seus dados, que são então validados e enviados ao banco de dados. Após o cadastro, o sistema gera um ID exclusivo para o usuário, que será utilizado junto com a senha para fazer login.

5. Login do Usuário: Foi desenvolvida a classe Login, que inclui uma interface gráfica para autenticar o usuário. Essa etapa é essencial, pois garante que apenas usuários registrados possam acessar o sistema e criar lotes ao

registrar análises. O login valida o ID e a senha no banco de dados, permitindo que, após a autenticação, o usuário prossiga para o registro de análises.

6. Registro de Análise: Desenvolvemos a classe RegistroAnalise, que captura os dados de uma análise realizada no laboratório. Essa classe recebe uma descrição da análise e permite o upload de um arquivo PDF. Em seguida, o arquivo é armazenado na tabela de registro de análises no banco de dados. Após a inserção bem-sucedida, o sistema devolve um número de lote, que serve como identificador único para o registro.

O pseudocódigo ajudou a guiar o desenvolvimento do código real, detalhando o fluxo da aplicação, a integração com o banco de dados, e os principais métodos e classes envolvidos no sistema.

3 CONSIDERAÇÕES FINAIS

O sistema desenvolvido para o NPCO2 da UFERSA mostrou-se eficaz no atendimento das necessidades de controle e gerenciamento de dados dos pesquisadores. A implementação em Java com suas bibliotecas proporcionou uma interface gráfica amigável e funcional, facilitando o uso por parte dos pesquisadores. Com isso, representando um avanço significativo na gestão das informações do laboratório. Através da utilização da Programação Orientada a Objetos (POO) em Java e a integração com um banco de dados relacional, foi possível criar uma solução eficiente e escalável, atendendo às necessidades específicas dos usuários. Além disso, a integração com o MySQL por meio da biblioteca JDBC facilitou a manipulação dos dados, assegurando que as operações de consulta, inserção e atualização fossem realizadas de maneira fluida e confiável. A utilização de pseudocódigo durante a fase de planejamento foi crucial, pois orientou a construção do código real e ajudou a esclarecer a lógica do sistema. Em suma, este trabalho não apenas atende às demandas do NPCO2, mas também estabelece uma base sólida para futuras expansões e melhorias no sistema.

REFERÊNCIAS

[1] SUN, Charles. *Java Programming with Swing*. New York: Pearson Education, 2015.

[2] UFERSA - Universidade Federal Rural do Semi-Árido. Núcleo de Pesquisa de Baixo Carbono (NPCO2). Disponível em: <www.ufersa.edu.br/npcO2>. Acesso em: 20 out. 2024.

[3] FARINELLI, Fernanda. Conceitos básicos de programação orientada a objetos. Instituto Federal Sudeste de Minas Gerais, 2007.

[4] RICARTE, Ivan Luiz Marques. Programação Orientada a Objetos: uma abordagem com Java. http://www. dca. fee. unicamp. br/cursos/PooJava/Aulas/poojava. pdf> Acesso em, v. 29, n. 10, p. 2014, 2001.

[5] MANOVICH, Lev. Banco de dados. Revista ECO-Pós, v. 18, n. 1, p. 7-26, 2015.

[6] INDRUSIAK, Leandro Soares. Linguagem java. Grupo JavaRS JUG Rio Grande do Sul, p. 19, 1996.