CENTRO UNIVERSITÁRIO SENAC SANTO AMARO

SISTEMA DE RECURSOS HUMANOS PARA GERENCIAMENTO DE FUNCIONÁRIOS

Integrantes:

Vitor Santana Sousa

João Pedro Barbosa da Silva

Nicolas Gonçalves Bábilas De Marco

Gabriel Henrique de Sousa

PROJETO INTEGRADOR

DESENVOLVIMENTO DE SISTEMAS ORIENTADO A OBJETOS

Professor Marcos de Arruda Monteiro

São Paulo

2024

Sumário

Descrição e Responsabilidades no Projeto

- **Vitor Santana Sousa**: Responsável pelo desenvolvimento principal da aplicação, incluindo o design, a implementação das funcionalidades, e a integração dos módulos para o gerenciamento eficiente dos dados dos funcionários.
- **Nicolas Gonçalves Bábilas De Marco**: Responsável pelo apoio no desenvolvimento e pela realização de testes na aplicação, contribuindo para a identificação e correção de bugs, e assegurando que todas as funcionalidades atendam aos requisitos do sistema.
- **Gabriel Henrique de Sousa**: Responsável pela produção do conteúdo da documentação, o que inclui a criação de descrições detalhadas das funcionalidades e a organização das informações técnicas sobre o sistema.
- **João Pedro Barbosa da Silva**: Apoio na produção do conteúdo e formatação da documentação, colaborando com Gabriel para garantir que a documentação seja clara, coerente e bem estruturada para facilitar a compreensão por todos os usuários e desenvolvedores.

Especificação Do Sistema

O sistema de Gerenciamento de Funcionários tem como objetivo otimizar e simplificar as atividades de administração de colaboradores no setor de Recursos Humanos (RH). Ele oferece funcionalidades completas para o cadastro, edição, exclusão, impressão e visualização das informações dos funcionários, facilitando o gerenciamento e o acesso a dados essenciais para o bom funcionamento da organização.

Planejamento

Etapas seguidas para o desenvolvimento do projeto:

1. **Definição dos requisitos**: Reunião inicial para identificar e documentar as necessidades do sistema.

2. **Desenvolvimento da aplicação**: Implementação das funcionalidades principais, incluindo o cadastro, edição, exclusão, impressão e visualização de funcionários.
3. **Testes e Validação**: Realização de testes unitários e de integração para verificar a consistência e o funcionamento adequado do sistema, corrigindo possíveis falhas e refinando o desempenho.
4. **Produção e Formatação da Documentação**: Criação de um manual de uso e documentação técnica do sistema, que abrange instruções de uso, estrutura do código e explicação dos métodos e funcionalidades implementadas.

Solução de Software

Cadastro de Funcionários

- **Campos**: Nome, CPF, RG, Cargo, Salário e Setor.
- **Validação**:
  - O CPF e RG são validados conforme padrões específicos.
  - O ID é gerado automaticamente no banco de dados (autoincremento).
- **Botões**:
  - **Limpar**: Limpa todos os campos de entrada.
  - **Cadastrar**: Cadastra o novo funcionário no banco de dados.

Alteração de Funcionários

- **Campos editáveis**: Nome, Cargo, Salário e Setor.
- **Campos não editáveis**: CPF e RG não podem ser alterados após o cadastro.
- **Botões**:
  - **Limpar**: Limpa todos os campos de entrada.
  - **Reverter**: Restaura os dados antigos do funcionário, exibindo os valores antes da modificação.
  - **Salvar**: Atualiza os dados do funcionário no banco de dados.

Exclusão de Funcionários

- **Campo**: Nome do funcionário.
- **Ação**: Confirmação da exclusão do funcionário.
- **Botão**:

- **Excluir**: Exclui o funcionário com o nome informado.

Impressão dos Funcionários

- **Ação**: Imprime todos os dados cadastrados de todos os funcionários.

Visualização dos Funcionários

- **Navegação**:
  - **Primeiro**: Exibe o primeiro funcionário da lista.
  - **Anterior**: Exibe o funcionário anterior da lista.
  - **Próximo**: Exibe o próximo funcionário da lista.
  - **Último**: Exibe o último funcionário da lista.

Banco de Dados

- **Conexão**:
  - Usando a classe `ConectarMySQL` para gerenciar a conexão com o banco de dados.
  - **Tabelas**: O sistema utiliza uma tabela `Funcionario` com os seguintes campos: `id`, `nome`, `cpf`, `rg`, `cargo`, `salario` e `setor`.

Interface

- A interface é construída utilizando o `JFrame` com uma organização em `GridLayout` para os campos e botões.
- O painel de entrada (`painelInputs`) e o painel de botões (`painelBotoes`) são criados separadamente e adicionados à interface principal.
- **Componente de Navegação**: Botões como "Primeiro", "Anterior", "Próximo", "Último" controlam a navegação pelos registros de funcionários.
- **Cadastro e Alteração**: Para adicionar ou modificar os dados, o sistema oferece janelas dedicadas (`JFrame`).
- O layout é personalizado com cores e fontes consistentes, com a utilização de `Color(65, 105, 225)` e botões com cores de fundo e texto configurados para melhorar a visibilidade e a experiência do usuário.

Manipulação de Dados

- **Adicionar Funcionário**: Ao cadastrar, os dados são validados (CPF, RG) e salvos no banco.
- **Alterar Funcionário**: Os dados modificados (exceto CPF e RG) são atualizados no banco.
- **Excluir Funcionário**: Exclui o funcionário do banco de dados baseado no nome informado.
- **Imprimir Funcionários**: Imprime todos os funcionários cadastrados no banco de dados.

Feedback ao Usuário

- O sistema oferece mensagens de erro ou sucesso através do `JOptionPane`, garantindo uma boa interação com o usuário.

Encapsulamento

Na classe `ConectarMySQL`:

- Os atributos (`url`, `username`, `password`, `con` e `rs`) são definidos como `private`, limitando o acesso direto a eles. Dessa forma, apenas métodos específicos da classe podem gerenciar a conexão com o banco de dados.
- A conexão é aberta e fechada por meio dos métodos `openDB()` e `closeDB()`, encapsulando a lógica de conexão e garantindo que o restante do código (como `TelaPrincipal`) não precise lidar diretamente com os detalhes internos de conexão.

Na classe `Funcionario`:

- Os atributos (`id`, `nome`, `cpf`, `rg`, `cargo`, `salario` e `setor`) são privados e acessíveis apenas por meio de métodos `get` e `set`, que controlam a leitura e a escrita desses valores.
- O método `toString()` permite a exibição dos dados do funcionário em um formato específico sem expor os detalhes internos de como esses dados são armazenados.

Na classe `TelaPrincipal`:

- Todos os atributos da interface gráfica e da lista de funcionários (`label`, `cadastroFrame`, `alteracaoFrame`, `painelInputs`, `txtId`, `funcionarios`, etc.) são privados, o que significa que eles só podem ser acessados dentro da própria classe. Isso evita que outras classes modifiquem diretamente esses atributos, garantindo que apenas os métodos internos da `TelaPrincipal` tenham controle sobre eles.
- Muitos métodos auxiliares, como `inicializarComponentes()`, `criarPainelInputs()`, `adicionarLabelsETextFields(JPanel panel)`, entre outros, são privados. Esses métodos ajudam a organizar o código e encapsulam a lógica de criação e manipulação dos componentes gráficos, mantendo a implementação detalhada oculta para classes externas. Isso melhora a legibilidade e facilita futuras manutenções ou modificações.
- O único ponto de entrada público é o método `main` e o construtor `TelaPrincipal`, que controla como a janela é instanciada e configurada. Ao encapsular as operações da interface gráfica e os dados, a `TelaPrincipal` protege seus estados internos de acessos indesejados, e classes externas não podem modificar a aparência ou o comportamento dos componentes gráficos diretamente.

Herança

O conceito de herança não é explicitamente utilizado nas classes fornecidas (`ConectarMySQL`, `Funcionario` e `TelaPrincipal`). Todas as classes seguem uma estrutura independente, sem uso de subclasses ou especializações. Cada classe contém funcionalidades específicas:

- **`ConectarMySQL`** é responsável pela conexão com o banco de dados e gerenciamento de recursos como `Connection`, `Statement` e `ResultSet`.
- **`Funcionario`** modela os dados de um funcionário, contendo atributos e métodos de acesso e formatação.
- **`TelaPrincipal`** representa a interface gráfica e manipula as interações com o usuário.

A herança poderia ser útil em situações como: gerenciar diferentes tipos de funcionários (temporário ou permanente); especializar as conexões do banco de dados.

Polimorfismo

Sobrecarga: Utilizada na classe `ConectarMySQL` nos métodos `closeDB` (existem três versões do método `closeDB`, cada uma com uma assinatura diferente), permitindo versatilidade ao liberar recursos de banco de dados.

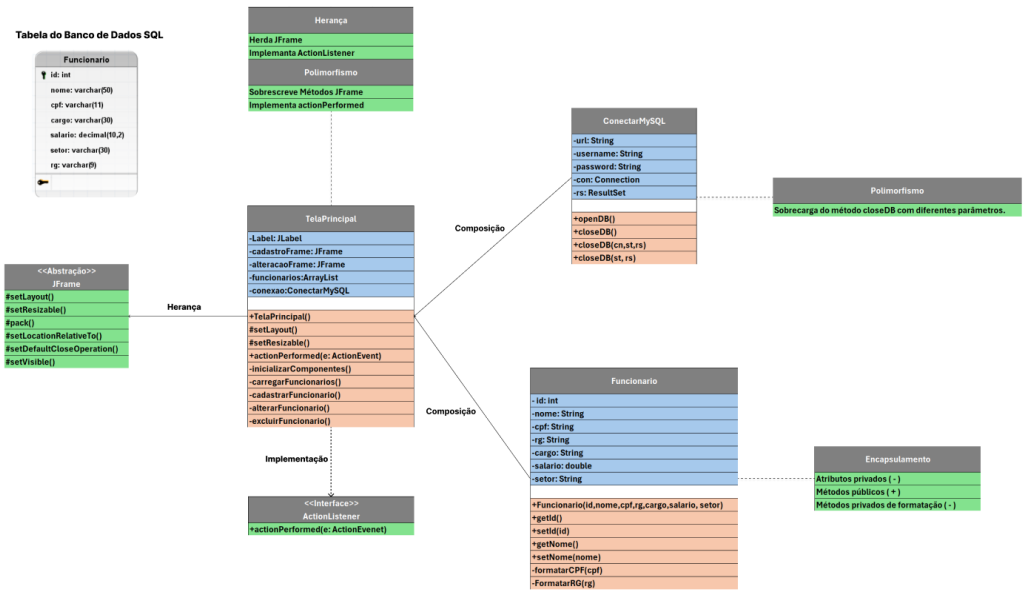
```
public void closeDB();
public void closeDB(Connection cn, Statement st, ResultSet rs2) throws SQLException;
public void closeDB(Statement st, ResultSet rs2) throws SQLException;
```

Sobrescrita: Aplicada na classe `TelaPrincipal` no método `actionPerformed` para definir respostas específicas a eventos de interface. Também usada na classe `Funcionario` para redefinir o método `toString`, oferecendo uma representação textual útil dos objetos `Funcionario`.

```
// Classe TelaPrincipal
@Override
public void actionPerformed(ActionEvent e) { ... }

// Classe Funcionario
@Override
public String toString() { ... }
```

Diagrama de Classe e Modelo de Dados



Conclusão

O Sistema de Gerenciamento de Funcionários desempenha um papel crucial na otimização e simplificação das atividades de administração de colaboradores no setor de Recursos Humanos (RH). Com um conjunto de funcionalidades robustas, como cadastro, edição, exclusão, impressão e visualização de informações, o sistema oferece uma solução eficaz para o gerenciamento de dados essenciais dos funcionários. Ao proporcionar uma plataforma centralizada e intuitiva, o sistema facilita o acesso rápido às informações dos colaboradores, contribuindo para a eficiência operacional e a organização dentro da empresa.

Combinando conceitos de orientação a objetos e uma interface simples e direta, o sistema resulta em uma plataforma robusta e eficiente, que melhora a organização interna e favorece a agilidade e precisão nas decisões estratégicas. Dessa forma, o uso dessa ferramenta não só torna a gestão de recursos humanos mais ágil e precisa, mas também favorece a melhoria contínua das práticas de RH, impulsionando a tomada de decisões mais informadas e eficazes.