UNIVERSIDADE PAULISTA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Felipe dos Santos

José Fernando da Silva Neto

Maria Clara Vieira

Natan Rodrigues dos Santos

SOFTWARE DE FOLHA DE PAGAMENTO

Campinas

2023

Felipe dos Santos – G703049
José Fernando da Silva Neto – N0389H4
Maria Clara Vieira – N7514B7
Natan Rodrigues do Santos – N7769F9

SOFTWARE DE FOLHA DE PAGAMENTO

Projeto Integrado Multidisciplinar – PIM, apresentado a UNIP – UNIVERSIDADE PAULISTA da cidade de Campinas, tal como exigência parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Ernesto Raschioto.

Campinas
2023

RESUMO

Este projeto possui a proposta de compreender os atuais desafios na geração de folha de pagamento para diversas empresas, visando realizar a automatização do processo para facilitar os cálculos e apresentação dos descritivos obtidos a partir dos resultados no holerite dos funcionários, para diminuir o trabalho manual da equipe de RH de realizar os cálculos individuais para cada empregado. Para o desenvolvimento do software foi utilizado de base o estudo de casos de uso que possam ajudar a desenvolver um sistema mais eficaz para os usuários, com uma interface fácil, interativa e intuitiva, focando nos pontos essenciais que os setores de RH das empresas necessitam para solucionar tais pontos e entregar um software de qualidade que possua boa elasticidade e escalabilidade para atender toda a demanda.

Palavras chaves: Folha de Pagamento. Software. Desenvolvimento.

ABSTRACT

This project aims to understand the current challenges in payroll generation for various companies, automating the process to facilitate calculations and presentation of descriptions obtained from the results on employees' pay slips, to reduce the team's manual work. HR department to perform individual calculations for each employee. To develop the software, we used the basis of a study of use cases that can help develop a more effective system for users, with an easy, interactive and intuitive interface, focusing on the essential points that the HR sectors of the companies involved to solve these points and deliver quality software that has good capacity and scalability to meet all demands.

Keywords: Payroll. Software. Development.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO

A folha de pagamento é um documento obrigatório na empresa como comprovante do pagamento do salário em conta ao funcionário, assim como é previsto na Lei n.º 8.212/91, da Consolidação da Legislação Previdenciária - CLP e na Consolidação das Leis do Trabalho - CLT pela Lei n.º 5.452/43. Algumas empresas optam pelo pagamento no último dia do mês, e outras optam pelo pagamento no quinto dia útil do mês, sendo esse o limite máximo preconizado pela lei. Em ambos os casos o funcionário possui a opção de receber uma porcentagem do salário no dia 20 ou 15 de cada mês, dado isso como adiantamento mensal, ou vale. E o conteúdo da folha de pagamento é justamente os proventos e descontos que um funcionário pode ter dentro da empresa de acordo com a convenção coletiva da cidade em que a empresa se situa e de acordo com as regras estabelecidas pela própria empresa do que será dado de benefício e o que será descontado da folha de pagamento do funcionário (RODRIGUES E BATISTA, 2016).

O processo de folha de pagamento é um processo complexo que demanda muita atenção na hora da sua geração pois trata-se do pagamento do salário na conta do trabalhador, e a prevenção de erros nessa situação evita tanto o desgaste entre empresa e funcionário, quanto uma imagem ruim por parte da empresa como fonte pagadora que não dá o mínimo aos seus colaboradores na hora da realização do pagamento.

Por esse motivo o sistema que gera essas informações deve ser o mais automatizado e eficiente possível e desenvolvido em métodos de prevenção e tratamento de erros que evitem conflitos para ambos os lados dentro da empresa. Deve ser um software seguro, de alta confiabilidade, usabilidade, intuitivo, fácil de se utilizar e que previna erros de sistema e de usuários. O objetivo desse relatório é apresentar o software desenvolvido de folha de pagamento automatizado para os setores de RH de diversas empresas, em que o software possui elasticidade e escalabilidade, se adequando a diversas empresas que precisam desse sistema, sendo o mais eficaz dentro do mercado de trabalho.

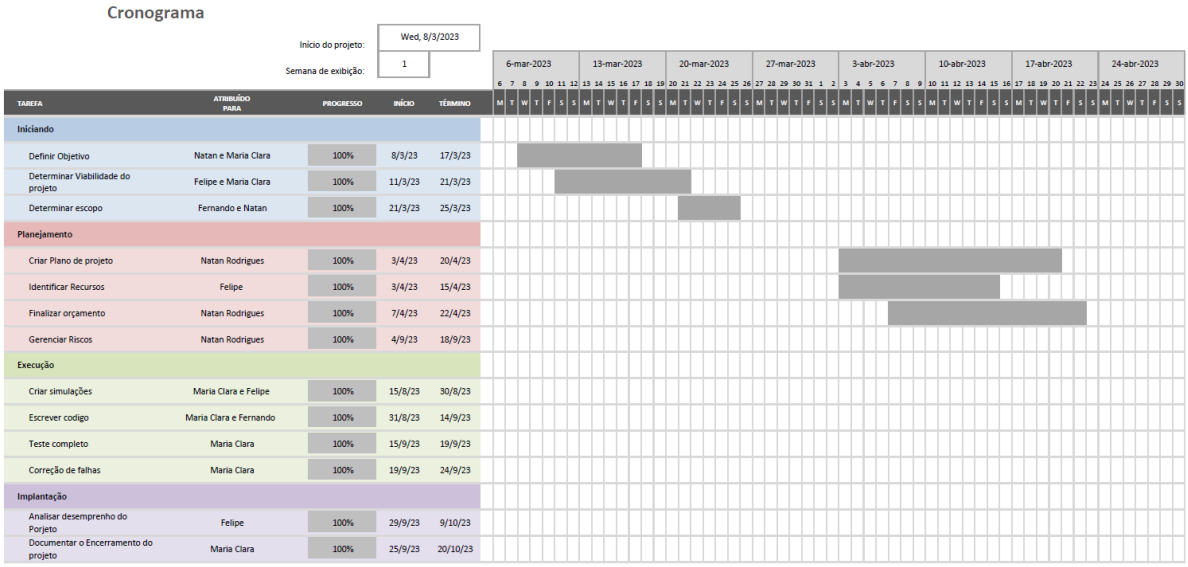
2. GERENCIAMENTO DE PROJETOS DE SOFTWARE

Nos tópicos a seguir serão apresentados o cronograma, custo, razoabilidade e riscos do desenvolvimento do projeto.

2.1. Cronograma

A Figura 1 – Diagrama de Gantt está representando o cronograma do desenvolvimento do projeto.

Figura 1 – Diagrama de Gantt



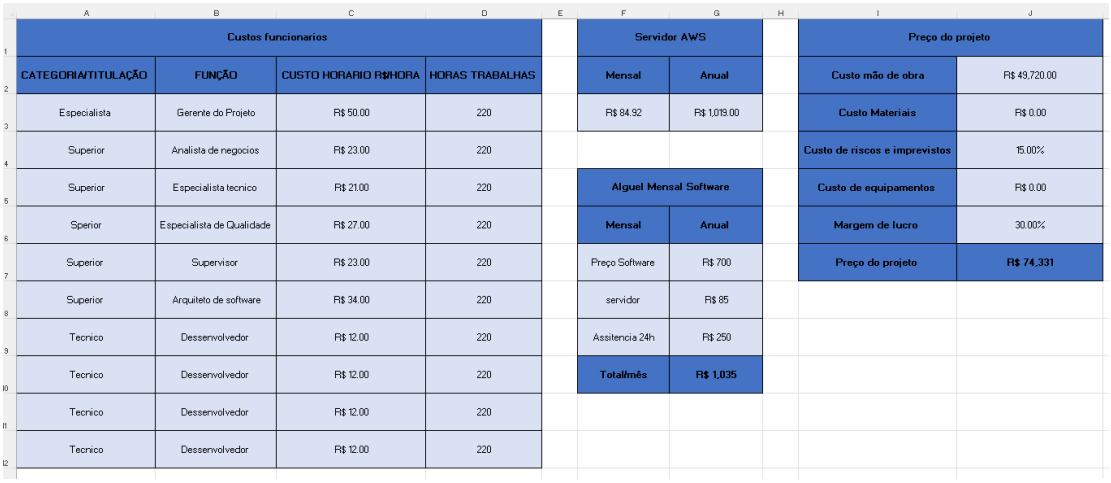
TAREFA	ATRIBUÍDO PARA	PROGRESSO	INÍCIO	TÉRMINO
Iniciando				
Definir Objetivo	Natan e Maria Clara	100%	8/3/23	17/3/23
Determinar Viabilidade do projeto	Felipe e Maria Clara	100%	11/3/23	21/3/23
Determinar escopo	Fernando e Natan	100%	21/3/23	25/3/23
Planejamento				
Criar Plano de projeto	Natan Rodrigues	100%	3/4/23	20/4/23
Identificar Recursos	Felipe	100%	3/4/23	15/4/23
Finalizar orçamento	Natan Rodrigues	100%	7/4/23	22/4/23
Gerenciar Riscos	Natan Rodrigues	100%	4/9/23	18/9/23
Execução				
Criar simulações	Maria Clara e Felipe	100%	15/8/23	30/8/23
Escrever codigo	Maria Clara e Fernando	100%	31/8/23	14/9/23
Teste completo	Maria Clara	100%	15/9/23	19/9/23
Correção de falhas	Maria Clara	100%	19/9/23	24/9/23
Implantação				
Analisar desemprenho do Porjeto	Felipe	100%	29/9/23	9/10/23
Documentar o Encerramento do projeto	Maria Clara	100%	25/9/23	20/10/23

Fonte: Própria.

2.2. Custos

A Figura 2 – Tabela de Custos está representando os custos para o desenvolvimento do projeto.

Figura 2 – Tabela de Custos



Custos funcionarios			
CATEGORIA/TITULAÇÃO	FUNÇÃO	CUSTO HORARIO R$/HORA	HORAS TRABALHAS
Especialista	Gerente do Projeto	R$ 50.00	220
Superior	Analista de negocios	R$ 23.00	220
Superior	Especialista tecnico	R$ 21.00	220
Sperior	Especialista de Qualidade	R$ 27.00	220
Superior	Supervisor	R$ 23.00	220
Superior	Arquiteto de software	R$ 34.00	220
Tecnico	Desenvolvedor	R$ 12.00	220
Tecnico	Desenvolvedor	R$ 12.00	220
Tecnico	Desenvolvedor	R$ 12.00	220
Tecnico	Desenvolvedor	R$ 12.00	220

Servidor AWS	
Mensal	Anual
R$ 84.92	R$ 1,019.00

Aluel Mensal Software	
Mensal	Anual
Preço Software	R$ 700
servidor	R$ 85
Assitencia 24h	R$ 250
Totalmês	R$ 1,035

Preço do projeto	
Custo mão de obra	R$ 49,720.00
Custo Materiais	R$ 0.00
Custo de riscos e imprevistos	15.00%
Custo de equipamentos	R$ 0.00
Margem de lucro	30.00%
Preço do projeto	R$ 74,331

Fonte: Própria.

2.3. Razoabilidade

O objetivo deste texto é avaliar a razoabilidade do projeto de implementação de um software com finalidade na automatização total da folha de pagamento. O projeto visa fornecer agilidade e facilidade e reduzir erros, e promovendo segurança de dados dos funcionários. A Implementação deste software se torna viável baseado em diversos fatores favoráveis, especialmente quando se considera o atual software incapaz de se integrar a outros softwares, causando trabalho manual excessivo, atrasos e erros. O novo software destaca-se por oferecer soluções a todos estes pontos, apresentando medidas que diminuem o trabalho manual e minimizando falhas.

Além disso, uma análise de custo foi realizada para o projeto, considerando a composição da equipe necessária para desenvolvimento. As estimativas foram fundamentadas em dados históricos de projetos similares resultando em custos foram razoáveis, com uma projeção de recuperação do investimento em um período aceitável.

A implementação deste software trará diversos benefícios para a empresa, será minimizado os riscos quanto a ocorrência de erros de digitação, cálculos incorretos, garantindo o pagamento de funcionários com precisão, contudo não podemos deixar de considerar os riscos envolvido, mas para cada risco foi tomada medidas de precaução, através de medidas de controles e capacitação das equipes.

Com base na análise realizada, fica evidente que o projeto de implementação do software de autorização da folha de pagamentos é uma decisão viável. Os benefícios gerados superam os custos, não resta dúvida que se deve avançar com o projeto. Recomenda-se acompanhando de perto sua implementação e resultados.

2.4. Riscos

Escopo: O risco de escopo é um risco crítico, onde se pode acontecer de não realizarmos aquilo que foi traçado como objetivo do projeto, por isso a definição dos requisitos e objetivos do projeto tem que ser de forma clara e concretizada, para ao longo do projeto não seja mudado o objetivo.

Definir o escopo de forma clara e objetiva e comunicar isso a todas as partes interessadas no início do projeto, e supervisionar todo o projeto ao longo de seu desenvolvimento, diminui a probabilidade de termos problemas como esse.

Baixo desempenho: Ter um baixo desempenho em equipes envolvidas no projeto é um risco que pode nos acarretar a vários outros problemas, onde o mesmo pode gerar, atraso nos prazos, sobrecargas em outras equipes e descumprimento do planejamento realizado. Esse risco é pode ser combatido com planejamento definido com clareza e compartilhado com as equipes, supervisões nos processos em tempo real e comunicação entre os membros da equipe, estas ações devem ser tomadas desde a fase de planejamento até a entrega final do projeto.

Custos Elevados: Custos elevados podem ser resultados de várias ações, colocando a parte financeira do projeto em situações críticas. Os custos devem ser elaborados no início do projeto e serem analisadas e revisadas ao longo de todo projeto.

Para não gerarmos este risco devemos manter estimativas bem definidas, não termos mudança de escopo de projeto, e sempre estar atento as variações e tendencias do mercado, pois se estes pontos não forem realizados de forma clara, podemos gerar outros riscos através dos custos elevados como o de Falta de recursos por problemas financeiros.

Falta de Recursos: Este risco pode ser caracterizado por falta de recursos humanos, falta de tempo, falta de dinheiro, ou falta de ferramentas necessárias causando um problema crucial no projeto, onde pode afetar na entrega do projeto. Este risco precisa ser tratado com cautela pelos gestores, pois eles são os

responsáveis por toda a aquisição de recursos, pois estes mesmos podem ser por falta de planejamento, por falta de estimativas custo ou por contratações erradas causando falta conhecimento técnico, para garantir a eliminação deste desse risco seria ideal que fosse realizado um Planejamento de Recursos necessários e ser revisado durante todo o projeto baseado no desenvolvimento do projeto.

Prazos curtos: O tempo é um dos principais aspectos, este deve ser definido não baseado apenas em entregas rápidas, mas deve ser realizadas estimativas generosas do tempo necessário para concluir uma tarefa, pois a falta de tempo por conta de prazos curtos acarreta vários problemas como entrega do projeto, e desempenhos gerais. O correto é que seja realizado um cronograma através de um Diagrama de Gantt e compartilhado com as equipes, e ser estipulados prazos que seja bom tanto para quem irá executar a atividade e para o gestor para que tenhamos um prazo que realmente são capazes de serem atendidos.

Falha técnica: Para um desenvolvimento de um projeto como este, envolve muita programação, e isto pode acarretar falhas de função do software, e isso pode levar tempo para ser corrigido e se deve ter m time preparado para eventos como este, por isso uns planejamentos de testes devem ser realizados e um plano de ação caso tenhamos uma falha inesperada

Segurança de Dados: Uma folha de pagamento carrega em si, todos os documentos particulares de uma pessoa, são dados que não podemos permitir que haja vazamentos para outros funcionários, este é um risco muito crítico, que coloca a qualidade do software em risco, por isso utilizar criptografias e autentificarão de usuário são essenciais.

Riscos legislatórios: Temos leis sendo mudadas constantemente e por conta disso podemos ter os requisitos do projeto alterado por conta de leis que surgem causando impacto na viabilidade do projeto e nos prazos, deste modo devemos estar atentos a todas as mudanças de leis.

Gestão de mudança: Todas as partes interessadas no projeto devem estar com todos os requisitos já acoplados no escopo do projeto, para que ao longo do projeto não seja necessárias solicitações de mudança, deste modo afetando o objetivo, o planejamento o foco do projeto, podendo causar atrasos, e causando outro risco chamado de desvio de escopo.

Partes interessadas: Quando partes interessadas não colaboram com foco e objetivo e suas participações são de baixa qualidade no projeto podemos causar

baixa qualidade em partes por conta de atitudes negativas, que levam ao um projeto sem total definição, e objetivo, desta forma deve-se garantir total participações de qualidade de todas as partes interessadas levando a umas combinações de participações que levam ao sucesso do projeto.

Implantação: A implantação do software é um dos momentos mais esperados, mais com mais necessidade de atenção, para que não tenhamos o risco de no ambiente em que será implementado o software não funcione, devemos criar um planejamento de testes parecidos com o que o software vai viver, e criar dicionários de erros e falhas, para que na implementação não tenha erro que desconhecidos levando a atraso nos prazos.

3. PROJETO DE SISTEMAS ORIENTADA A OBJETOS

O tópico a seguir descreve e detalha os casos de uso do sistema desenvolvido.

3.1. Diagrama de Casos de Uso

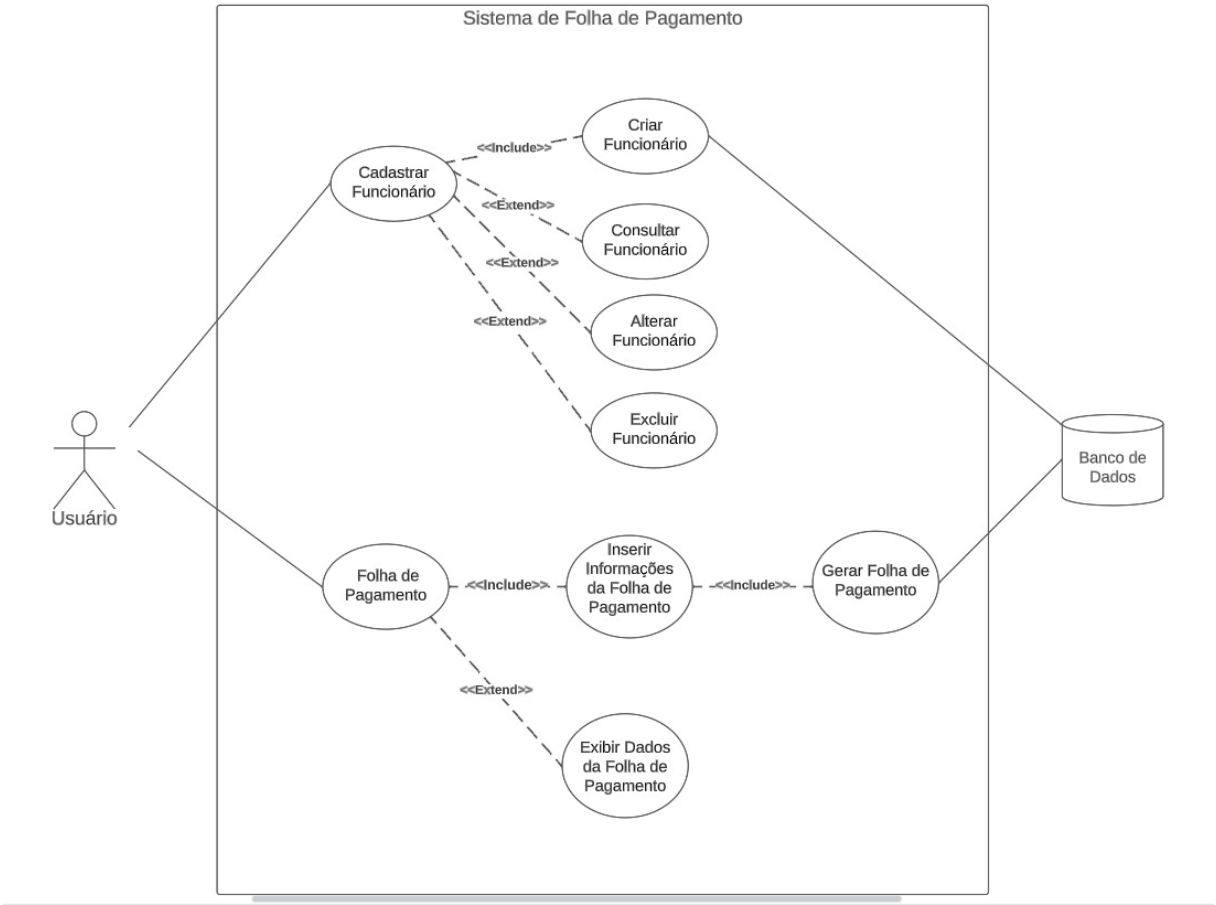
Segundo COSTA (2001), a UML é um conjunto de diagramas que representam toda a estrutura do sistema a ser desenvolvido sob análise do desenvolvedor/analista. É uma apresentação visual das diferentes formas e cenários possíveis de uso de um sistema. O diagrama ilustrado nesse relatório é o Digrama de Casos de Uso, que ilustra como um usuário realizará ações e irá interagir com um sistema específico, como por exemplo um site ou um aplicativo.

O sistema de folha pagamento criado permite ao usuário o cadastramento de funcionário com informações como Nome, CPF, Endereço, Cargo, Salário etc. Feito o cadastramento do funcionário suas informações são salvas no banco de dados onde é possível realizar operações como consultar informações do funcionário, alterar informações do funcionário e a exclusão do funcionário.

O funcionário estando cadastrado no sistema é permitido gerar sua folha de pagamento de forma automática, inserindo informações como o CPF, dias trabalhados, faltas etc.

Conforme Figura 3 – Diagrama de Casos de Uso, é possível a visualização do caso de uso do sistema de folha de pagamento.

Figura 3 – Diagrama de Casos de Uso



Fonte: Própria.

O relatório a seguir busca especificar todo o caso de uso “Cadastrar Funcionário” mostrando o seu fluxo principal, sub-fluxo, fluxo de exceção, fluxos alternativos, regras de negócios e validações.

Nome do caso de uso:	Cadastrar Funcionário
Ator principal:	Usuário
Ator Secundário:	N/A
Descrição:	O caso de uso Cadastrar Funcionário permite a criação um funcionário e salvar suas informações no banco de dados, além de conseguir consultar seus dados, alterar e realizar a exclusão.
Pré-Condições:	Informar dados válidos do Funcionário.
Pós-Condições:	Manter suas informações salvas no

	banco de dados para futuras consultas, alterações ou exclusões.
Fluxo Principal	
Ações do Ator: 1. O sistema deve permitir que o usuário insira os dados pessoais (CPF, Nome, Endereço etc.) do funcionário. 2. O sistema deve permitir que o usuário insira os dados sobre o cargo (Função, Salário etc.) do funcionário. 3. O sistema deve permitir que o usuário insira as informações bancárias (Banco, Agência, Conta) do colaborador.	**Ações do Sistema:** 4. Após as informações serem inseridas o sistema deverá armazenar as informações no banco de dados. 5. O sistema executará o sub-fluxo referente ao tipo de operação escolhida (Incluir, Alterar, Excluir ou Consultar).
Sub-Fluxo Consultar	
Ações do Ator: 6. O sistema deve permitir que o usuário filtre o funcionário pelo CPF. 7. O sistema deve permitir que o usuário selecione a opção buscar.	**Ações do sistema:** 8. O sistema deverá buscar o CPF digitado no banco de dados. 9. O sistema deverá apresentar na tela as informações do funcionário.
Sub-Fluxo Incluir	
Ações do Ator: 10. O sistema deve permitir que o usuário preencha informações pessoais, informações sobre o cargo e dados bancários do funcionário. 11. O sistema deve permitir o usuário a selecionar a opção de salvar.	**Ações do sistema:** 12. O sistema deverá salvar as informações no banco de dados. 13. O sistema deverá apresentar uma mensagem informando que a operação foi bem-sucedida.
Sub-Fluxo Alterar	
Ações do Ator: 14. O sistema deve permitir que o usuário filtre o funcionário pelo CPF. 15. O sistema deve permitir que o	**Ações do sistema:** 16. O sistema deverá apresentar na tela as informações do funcionário. 19. O sistema deverá salvar as novas

usuário selecione a opção buscar. 17. O sistema deve permitir que o usuário altere os dados do funcionário. 18. O sistema deve permitir que o usuário selecione a opção de salvar.	informações no banco de dados.
Sub-Fluxo Excluir	
Ações do Ator: 20. O sistema deve permitir que o usuário filtro o funcionário pelo CPF. 21. O sistema deve permitir que o usuário selecione a opção excluir. 23. O sistema deve permitir que o usuário confirme a informação de exclusão.	**Ações do sistema:** 22. O sistema deverá apresentar uma mensagem na tela de confirmação da exclusão. 24. O sistema deverá excluir as informações do funcionário do banco de dados.
Fluxos de Exceção:	1. O sistema deverá exibir uma mensagem de erro caso tente criar um funcionário e esqueça o preenchimento de algum campo. 2. O sistema deverá exibir uma mensagem de erro caso tente ser realizado a consulta de um funcionário que não esteja cadastrado no banco de dados. 3. O sistema deverá exibir uma mensagem de erro caso seja inserido algum caractere inválido. (Exemplo: letra no CPF). 4. O sistema deverá exibir uma mensagem de erro caso tente ser feito a exclusão de um CPF que não exista no banco de dados.
Fluxos alternativos:	1. O sistema deve permitir que o usuário

	cancele alguma ação e retorne a tela inicial do sistema.
Regras de negócios:	1. Não pode ser cadastrado dois funcionários com o mesmo CPF. 2. O CPF do funcionário deve ser único e válido. 3. Os dados bancários do funcionário devem ser de uma conta ativa. 4. Para consultar informações do funcionário, deve ser informado o seu CPF completo.
Validações:	1. O CPF do funcionário deve ter no máximo 11 dígitos. 2. A data de nascimento do colaborador não pode ter posterior a data atual. 3. O campo Nome deve ser preenchido por pelo menos duas palavras. 4. Todos os campos devem ser preenchidos no momento do cadastro do funcionário.

4. DESENVOLVIMENTO DE SOFTWARE PARA INTERNET

No tópico a seguir será apresentada as telas codificadas em HTML, CSS e JavaScript.

4.1. Codificação FRONT-END

Figura 4 – Interface

Fonte: Própria.

Figura 5 – Código HTML: Parte 1

```
index.html > html > head > title
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Nérus</title>
</head>
<body>
    <header>
        <h2 class="logo">Nérus</h2>
        <nav class="navigation">
            <a href="#">Home</a>
            <a href="#">Folha</a>
            <a href="#">Cadastro</a>
            <button class="btmLogin">Login</button>
        </nav>
    </header>
    <div class="info">
        <h2>Bem-vindo à Nérus: Seu Parceiro em Recursos Humanos</h2>
        <p class="home">Na Nérus, acreditamos que as pessoas são o ativo mais valioso de qualquer empresa. Como uma empresa de Recursos Humanos comprometida em tornar o s
        </p>
            <h2>Nossos Valores Fundamentais:</h2>
            <ul>
            <li>Excelência: Na Nérus, nos esforçamos constantemente para alcançar a excelência em tudo o que fazemos. Buscamos a qualidade em nossos serviços e soluções,

            <li>Integridade: Agimos com integridade e ética em todos os aspectos do nosso negócio. A confiança é a base das relações que construímos com nossos clientes e

            <li>Colaboração: Acreditamos no poder da colaboração e no trabalho em equipe. Trabalhamos em estreita parceria com nossos clientes para entender suas necessid

            <li>Inovação: O mundo dos negócios está em constante evolução, e na Nérus, abraçamos a inovação como um meio de enfrentar desafios e aproveitar oportunidades.

            <li>Compromisso com o Cliente: A satisfação dos nossos clientes é a nossa prioridade. Estamos comprometidos em fornecer soluções personalizadas que atendam às

            <li>Desenvolvimento Pessoal: Acreditamos no desenvolvimento contínuo e na capacitação dos profissionais. Estamos comprometidos em ajudar os funcionários a cre
        </ul><br>
```

Fonte: Própria.

Figura 6 – Código HTML: Parte 2

```
        </ul><br>
            <p>Na Nérus, nosso objetivo é fazer com que sua equipe seja a melhor versão de si mesma, proporcionando soluções de RH excepcionais e orientadas para o sucess

        </div>
<div class="wrapper">
    <span class="icon-close"><ion-icon name="close-circle-outline"></ion-icon></span>
    <div class="form-box login">
        <h2>Login</h2>
        <form action="#">
            <div class="input-box">
                <span class="icon"><ion-icon name="mail"></ion-icon></span>
                <input type="email" required>
                <label>Email</label>
            </div>
            <div class="input-box">
                <span class="icon"><ion-icon name="lock-closed"></ion-icon></span>
                <input type="password" required>
                <label>Senha</label>
            </div>
            <div class="remember-forgot">
                <label><input type="checkbox">
                Lembrar</label>
                <a href="#">Esqueci minha senha!</a>
            </div>
            <button type="submit" class="btn">Entrar</button>
            <div class="login-register">
                <p>Não é registrado?<a href="#" class="register-link">Registre</a></p>
            </div>
        </form>
    </div>
```

Fonte: Própria.

Figura 7 – Código HTML: Parte 3

```
    <div class="form-box register">
        <h2>Cadastro</h2>
        <form action="#">
            <div class="input-box">
                <span class="icon"><ion-icon name="person"></ion-icon></span>
                <input type="text" required>
                <label>Usuario</label>
            </div>
            <div class="input-box">
                <span class="icon"><ion-icon name="mail"></ion-icon></span>
                <input type="email" required>
                <label>Email</label>
            </div>
            <div class="input-box">
                <span class="icon"><ion-icon name="lock-closed"></ion-icon></span>
                <input type="password" required>
                <label>Senha</label>
            </div>
            <div class="remember-forgot">
                <label><input type="checkbox">
                Aceitar termos!</label>
            </div>
            <button type="submit" class="btn">Registar</button>
            <div class="login-register">
                <p>Ja sou cadastrado!<a href="#" class="login-link">Login</a></p>
            </div>
        </form>
    </div>
</div>

    <script src="script.js"></script>
    <script type="module" src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
<script nomodule src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.js"></script>
</body>
</html>
```

Fonte: Própria.

Figura 8 – Código CSS: Parte 1

```css
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: sans-serif;
}

body{
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background: rgb(216, 216, 216);
    background-size: cover;
    background-position: center;
}

header{
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    padding: 20px 100px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    z-index: 99;
}

.logo{
    font-size: 2em;
    color: #000000;
    user-select: none;
}

```

Fonte: Própria.

Figura 9 – Código CSS: Parte 2

```css
.navigation a {
    position: relative;
    font-size: 1.1em;
    color: #000;
    text-decoration: none;
    font-weight: 500;
    margin-left: 40px;
}

.navigation a::after{
    content: '';
    position: absolute;
    left: 0;
    bottom: -6px;
    width: 100%;
    height: 3px;
    background: #000;
    border-radius: 5px;
    transform-origin: right;
    transform: scaleX(0);
    transition: transform .5s;
}

.navigation a:hover::after{
    transform: scaleX(1);
    transform-origin: left;
}

.navigation .btmLogin{
    width: 130px;
    height: 50px;
    background: transparent;
    border: 2px solid #000;
    outline: none;
    border-radius: 6px;
    cursor: pointer;
```

Fonte: Própria.

Figura 10 – Código CSS: Parte 3

```
    font-size: 1.1em;
    color: #000;
    font-weight: 500;
    margin-left: 40px;
    transition: .5s;
}

.navigation .btmLogin:hover{
    background: #000;
    color: #fff;
}
.info{
    width: 100%;
    background-color: white;
    border-radius: 10px;
    padding: 25px;
}
p.home{
    text-align: justify;
}
.wrapper{
    position: relative;
    width: 400px;
    height: 440px;
    background: transparent;
    border: 2px solid rgba(255,255,255, .5);
    border-radius: 20px;
    backdrop-filter: blur(20px);
    box-shadow: rgba(0, 0, 0, .5);
    display: flex;
    justify-content: center;
    align-items: center;
    overflow: hidden;
    transform: scale(0);
    transition: transform .3s ease, height .2s ease;
}
```

Fonte: Própria.

Figura 11 – Código CSS: Parte 4

```
.wrapper.active-popup{
    transform: scale(1);
}

.wrapper.active{
    height: 520px;
}

.wrapper .form-box{
    width: 100%;
    padding: 40px;
}

.wrapper .form-box.login{
    transition: transform .18s ease;
    transform: translateX(0);
}

.wrapper.active .form-box.login{
    transition: none;
    transform: translateX(-400px);
}

.wrapper .form-box.register{
    position: absolute;
    transition: none;
    transform: translateX(400px);
}

.wrapper.active .form-box.register{
    transition: transform .18s ease;
    transform: translateX(0);
}
```

Fonte: Própria.

Figura 12 – Código CSS: Parte 5

```css
.wrapper .icon-close{
    position: absolute;
    top: 0;
    right: 0;
    width: 35px;
    height: 35px;
    font-size: 2em;
    color: #282828;
    display: flex;
    justify-content: center;
    align-items: center;
    cursor: pointer;
    z-index: 1;
    border-radius: 45px;
}

.form-box h2{
    font-size: 2em;
    color: black;
    text-align: center;
}

.input-box{
    position: relative;
    width: 100%;
    height: 50px;
    border-bottom: 2px solid black;
    margin: 30px 0;
}

.input-box label{
    position: absolute;
    top: 50%;
    left: 5px;
    transform: translateY(-50%);
    font-size: 1em;
```

Fonte: Própria.

Figura 13 – Código CSS: Parte 6

```css
    font-size: 1em;
    color: #282828;
    font-weight: 500;
    pointer-events: none;
    transition: .5s;
}

.input-box input:focus~label,
.input-box input:valid~label{
    top: -5px;
}

.input-box input{
    width: 100%;
    height: 100%;
    background: transparent;
    border: none;
    outline: none;
    font-size: 1em;
    color: #282828;
    font-weight: 600;
    padding: 0 35px 0 5px;
}

.input-box .icon{
    position: absolute;
    right: 8px;
    font-size: 1.2em;
    color: #282828;
    line-height: 57px;
}

.remember-forgot{
    font-size: .9em;
    color: #282828;
    font-weight: 500;
```

Fonte: Própria.

Figura 14 – Código CSS: Parte 7

```css
.remember-forgot{
    font-size: .9em;
    color: #282828;
    font-weight: 500;
    margin: -15px 0 15px;
    display: flex;
    justify-content: space-between;
}

.remember-forgot label input{
    accent-color: #14002b;
    margin-right: 3px;
}

.remember-forgot a{
    color: #282828;
    text-decoration: none;
}

.remember-forgot a:hover{
    text-decoration: underline;
}

.btn{
    width: 100%;
    height: 45px;
    background: #282828;
    border: none;
    outline: none;
    border-radius: 6px;
    cursor: pointer;
    font-size: 1em;
    color: #fff;
    font-weight: 500;
}
```

Fonte: Própria.

Figura 15 – Código CSS: Parte 8

```css

.login-register{
    font-size: .9em;
    color: #282828;
    text-align: center;
    font-weight: 500;
    margin: 25px 0 10px;
}

.login-register p a{
    color: #282828;
    text-decoration: none;
    font-weight: 600;
}

.login-register p a:hover{
    text-decoration: underline;
}

```

Figura 16 – Código JavaScript

```
JS script.js > ...
1   const wrapper = document.querySelector('.wrapper')
2   const loginLink = document.querySelector('.login-link')
3   const registerLink = document.querySelector('.register-link')
4   const btnLogin = document.querySelector('.btnLogin')
5   const iconClose = document.querySelector('.icon-close')
6
7   registerLink.addEventListener('click', ()=> {
8       wrapper.classList.add('active')
9   })
10
11  loginLink.addEventListener('click', ()=> {
12      wrapper.classList.remove('active')
13  })
14
15  btnLogin.addEventListener('click', ()=> {
16      wrapper.classList.add('active-popup')
17  })
18
19  iconClose.addEventListener('click', ()=> {
20      wrapper.classList.remove('active-popup')
21  })
22
```

Fonte: Própria.

5. PROGRAMAÇÃO ORIENTADA A OBJETOS

Código-fonte das classes MODEL, VIEW, CONTROLER E DAO do sistema desenvolvido e documentado nesse relatório, apresentados no APÊNDICE A.

6. CONSIDERAÇÕES FINAIS

Neste trabalho, foi documentado o processo de desenvolvimento de um software de folha de pagamento automatizado para uma empresa de Recursos Humanos/Departamento Pessoal.

O relatório foi desenvolvido sob um olhar de BI (Business Inteligence) e mostrou que o processo de desenvolvimento de um sistema de software tende a ser bem estruturado, capaz de atender aos requisitos do cliente em todas as suas especificações. O sistema desenvolvido busca automatizar ao máximo as atividades relacionadas à folha de pagamento, adotando métodos de prevenção e tratamento de erros, visando garantir a segurança, confiabilidade e usabilidade do software.

Dessa forma, o presente trabalho contribui para a documentação do código-fonte de um possível protótipo que leva ao desenvolvimento de um sistema de folha de pagamento automatizado para diversas empresas de Recursos Humanos/Departamento Pessoal.

REFERÊNCIAS BIBLIOGRÁFICAS

BRASIL. Ministério da Cidadania. **Lei Geral de Proteção de Dados Pessoais (LGPD)**. Brasília, DF: Ministério da Cidadania. Disponível em: https://www.gov.br/cidadania/pt-br/acesso-a-informacao/lgpd. Acesso em: 01 out. 2022.

COSTA, A. C. A aplicação de modelagem unificada (UML) para o suporte ao projeto de sistemas computacionais dentro de um modelo de referência. **Gestão e Produção,** v.8, n.1, p.19-36, abr. 2001.

RODRIGUES. M. R. G., BATISTA, F. J. **Estudo da folha de pagamento: principais proventos e descontos.** ANAIS - Seminário de Estágio Supervisionado do Campus Anápolis de CSEHUEG: as decisões nas políticas públicas nacionais, estaduais e institucionais com reflexos na formação profissional. Goiás. nov. 2016.

APÊNDICE A – CÓDIGO-FONTE DO SISTEMA

1. Classes Funcionarios

```
public partial class CadastrarFuncionarios : Form
{

    Funcionarios funcionarios;
    CtrFuncionarios ctrFuncionarios;

    1 reference
    public CadastrarFuncionarios()
    {
        InitializeComponent();
        txtCpf.TabIndex = 0;
        funcionarios = new Funcionarios();
        ctrFuncionarios = new CtrFuncionarios();
    }

    1 reference
    private void btnSalvar_Click(object sender, EventArgs e)
    {
        if (ValidarCampos())
        {
            funcionarios.Nome = txtNome.Text;
            funcionarios.CPF = txtCpf.Text;
            funcionarios.RG = txtRg.Text;
            funcionarios.EstadoCivil = cbEstadoCivil.SelectedItem.ToString();
            funcionarios.DataNascimento = txtDtNascimento.Text;
            funcionarios.EnderecoCompleto = txtEnderecoCompleto.Text;
            funcionarios.PIS = txtPis.Text;
            funcionarios.TituloEleitor = txtTituloEleitor.Text;
            funcionarios.CTPS = txtCtps.Text;
            funcionarios.Reservista = txtReservista.Text;
            funcionarios.Cargo = txtCargo.Text;
            funcionarios.SalarioBase = Convert.ToDecimal(txtSalario.Text);
            funcionarios.IndicadorPlanoSaude = cbPs.SelectedItem.ToString();
            funcionarios.IndicadorPlanoOdonto = cbPo.SelectedItem.ToString();
            funcionarios.IndicadorValeTransporte = cbVt.SelectedItem.ToString();
            funcionarios.IndicadorValeRefeicao = cbVr.SelectedItem.ToString();
            funcionarios.IndicadorValeAlimentacao = cbVa.SelectedItem.ToString();
            funcionarios.Banco = txtBanco.Text;
            funcionarios.Agencia = txtAgencia.Text;
            funcionarios.Conta = txtConta.Text;
            ctrFuncionarios.createFuncionarios(funcionarios);
            LimparDados();
        }
        else
        {
            MessageBox.Show("Preencher todos os campos obrigatoriamente.");
        }
    }
```

```
1 reference
private void btnBuscar_Click(object sender, EventArgs e)
{
    if (txtCpf.MaskCompleted)
    {
        funcionarios.CPF = txtCpf.Text;
        bool cpfExists = ctrFuncionarios.validarFuncionario(funcionarios);

        if (cpfExists)
        {
            funcionarios.CPF = txtCpf.Text;
            ctrFuncionarios.readFuncionarios(funcionarios);
            txtNome.Text = funcionarios.Nome;
            txtCpf.Text = funcionarios.CPF;
            txtRg.Text = funcionarios.RG;
            cbEstadoCivil.SelectedItem = funcionarios.EstadoCivil;
            txtDtNascimento.Text = funcionarios.DataNascimento;
            txtEnderecoCompleto.Text = funcionarios.EnderecoCompleto;
            txtPis.Text = funcionarios.PIS;
            txtTituloEleitor.Text = funcionarios.TituloEleitor;
            txtCtps.Text = funcionarios.CTPS;
            txtReservista.Text = funcionarios.Reservista;
            txtCargo.Text = funcionarios.Cargo;
            txtSalario.Text = Convert.ToString(funcionarios.SalarioBase);
            cbPs.SelectedItem = funcionarios.IndicadorPlanoSaude;
            cbPo.SelectedItem = funcionarios.IndicadorPlanoOdonto;
            cbVt.SelectedItem = funcionarios.IndicadorValeTransporte;
            cbVr.SelectedItem = funcionarios.IndicadorValeRefeicao;
            cbVa.SelectedItem = funcionarios.IndicadorValeAlimentacao;
            txtBanco.Text = funcionarios.Banco;
            txtAgencia.Text = funcionarios.Agencia;
            txtConta.Text = funcionarios.Conta;
        }
        else
        {
            MessageBox.Show("CPF não encontrado no banco de dados.");
        }
    }
    else
        MessageBox.Show("CPF inválido");
}
```

```
1 reference
private void btnExcluir_Click(object sender, EventArgs e)
{
    if (txtCpf.MaskCompleted)
    {
        funcionarios.CPF = txtCpf.Text;
        bool cpfExists = ctrFuncionarios.validarFuncionario(funcionarios);

        if (cpfExists)
        {
            DialogResult result = MessageBox.Show("Tem certeza de que deseja excluir este funcionário?", "Confirmação de Exclusão", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

            if (result == DialogResult.Yes)
            {
                ctrFuncionarios.deleteFuncionarios(funcionarios);
                LimparDados();
            }
            else
            {
                LimparDados();
            }
        }
        else
        {
            MessageBox.Show("CPF não encontrado no banco de dados.");
        }
    }
    else
        MessageBox.Show("CPF inválido");
}
```

```
1 reference
private void btnAtualizar_Click(object sender, EventArgs e)
{
    if (ValidarCampos())
    {
        funcionarios.CPF = txtCpf.Text;
        bool cpfExists = ctrFuncionarios.validarFuncionario(funcionarios);

        if (cpfExists)
        {
            funcionarios.Nome = txtNome.Text;
            funcionarios.RG = txtRg.Text;
            funcionarios.EstadoCivil = cbEstadoCivil.SelectedItem.ToString();
            funcionarios.DataNascimento = txtDtNascimento.Text;
            funcionarios.EnderecoCompleto = txtEnderecoCompleto.Text;
            funcionarios.PIS = txtPis.Text;
            funcionarios.TituloEleitor = txtTituloEleitor.Text;
            funcionarios.CTPS = txtCtps.Text;
            funcionarios.Reservista = txtReservista.Text;
            funcionarios.Cargo = txtCargo.Text;
            funcionarios.SalarioBase = Convert.ToDecimal(txtSalario.Text);
            funcionarios.IndicadorPlanoSaude = cbPs.SelectedItem.ToString();
            funcionarios.IndicadorPlanoOdonto = cbPo.SelectedItem.ToString();
            funcionarios.IndicadorValeTransporte = cbVt.SelectedItem.ToString();
            funcionarios.IndicadorValeRefeicao = cbVr.SelectedItem.ToString();
            funcionarios.IndicadorValeAlimentacao = cbVa.SelectedItem.ToString();
            funcionarios.Banco = txtBanco.Text;
            funcionarios.Agencia = txtAgencia.Text;
            funcionarios.Conta = txtConta.Text;
            ctrFuncionarios.updateFuncionarios(funcionarios);
            LimparDados();
        }
        else
        {
            MessageBox.Show("CPF não encontrado no banco de dados.");
        }
    }
    else
    {
        MessageBox.Show("Preencher todos os campos obrigatoriamente.");
    }
}

1 reference
private void btnLimpar_Click(object sender, EventArgs e)
{
    LimparDados();
}
```

```
5 references
private void LimparDados()
{
    txtNome.Text = "";
    txtCpf.Text = "";
    txtRg.Text = "";
    cbEstadoCivil.SelectedItem = null;
    txtDtNascimento.Text = "";
    txtEnderecoCompleto.Text = "";
    txtPis.Text = "";
    txtTituloEleitor.Text = "";
    txtCtps.Text = "";
    txtReservista.Text = "";
    txtCargo.Text = "";
    txtSalario.Text = "";
    cbPs.SelectedItem = null;
    cbPo.SelectedItem = null;
    cbVt.SelectedItem = null;
    cbVr.SelectedItem = null;
    cbVa.SelectedItem = null;
    txtBanco.Text = "";
    txtAgencia.Text = "";
    txtConta.Text = "";
    txtCpf.Focus();
}

2 references
private bool ValidarCampos()
{
    if (txtNome.Text != "" && txtCpf.Text != "" && txtRg.Text != "" && cbEstadoCivil.SelectedItem != null && txtDtNascimento.Text != "" &&
        txtEnderecoCompleto.Text != "" && txtPis.Text != "" && txtTituloEleitor.Text != "" && txtCtps.Text != "" && txtReservista.Text != "" &&
        txtCargo.Text != "" && txtCargo.Text != "" && txtSalario.Text != "" && cbPs.SelectedItem != null && cbPo.SelectedItem != null &&
        cbVt.SelectedItem != null && cbVr.SelectedItem != null && cbVa.SelectedItem != null && txtBanco.Text != "" && txtAgencia.Text != "" && txtConta.Text != "")
        return true;
    else return false;
}

1 reference
private void btnFechar_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Deseja realmente fechar o formulário?", "Confirmação de Fechamento", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (result == DialogResult.Yes)
    {
        this.Close();
    }
}
```

```
namespace PIM_IV.Models
{
    13 references
    class Funcionarios
    {
        6 references
        public string Nome { get; set; }
        12 references
        public string CPF { get; set; }
        6 references
        public string RG { get; set; }
        6 references
        public string DataNascimento { get; set; }
        6 references
        public string EstadoCivil { get; set; }
        6 references
        public string EnderecoCompleto { get; set; }
        6 references
        public string PIS { get; set; }
        6 references
        public string TituloEleitor { get; set; }
        6 references
        public string CTPS { get; set; }
        6 references
        public string Reservista { get; set; }
        6 references
        public string Cargo { get; set; }
        6 references
        public decimal SalarioBase { get; set; }
        6 references
        public string IndicadorPlanoSaude { get; set; }
        6 references
        public string IndicadorPlanoOdonto { get; set; }
        6 references
        public string IndicadorValeTransporte { get; set; }
        6 references
        public string IndicadorValeRefeicao { get; set; }
        6 references
        public string IndicadorValeAlimentacao { get; set; }
        6 references
        public string Banco { get; set; }
        6 references
        public string Agencia { get; set; }
        6 references
        public string Conta { get; set; }
    }
}
```

```
namespace PIM_IV.Controllers
{
    3 references
    class CtrFuncionarios
    {
        Dao_Funcionarios dao_Funcionarios;

        1 reference
        public CtrFuncionarios()
        {
            dao_Funcionarios = new Dao_Funcionarios();
        }


        1 reference
        public void createFuncionarios(Funcionarios funcionarios)
        {
            dao_Funcionarios.create(funcionarios);
        }

        1 reference
        public void readFuncionarios(Funcionarios funcionarios)
        {
            dao_Funcionarios.read(funcionarios);
        }

        1 reference
        public void deleteFuncionarios(Funcionarios funcionarios)
        {
            dao_Funcionarios.delete(funcionarios);
        }

        1 reference
        public void updateFuncionarios(Funcionarios funcionarios)
        {
            dao_Funcionarios.update(funcionarios);
        }

        3 references
        public bool validarFuncionario(Funcionarios funcionarios)
        {
            return dao_Funcionarios.validarFuncionario(funcionarios);
        }
    }
}
```

```
2 references
class Dao_Funcionarios
{
    SqlConnection connection = new SqlConnection(@"Data Source=VIEIRMC-W11ASUS\SQLSERVER2022;Initial Catalog=PIM_IV;Integrated Security=True;");

    1 reference
    public void create(Funcionarios funcionarios)
    {
        try
        {
            string commandSql = "INSERT INTO Funcionarios(Nome,CPF,RG,DataNascimento,EstadoCivil,EnderecoCompleto,PIS,TituloEleitor,CTPS,Reservista,Cargo,SalarioBase,IndicadorPlanoSaude,IndicadorPlanoOdonto,IndicadorValeTransporte,IndicadorValeRefeicao,
            SqlCommand command = new SqlCommand(commandSql, connection);

            connection.Open();

            command.Parameters.AddWithValue("@Nome", funcionarios.Nome);
            command.Parameters.AddWithValue("@CPF", funcionarios.CPF);
            command.Parameters.AddWithValue("@RG", funcionarios.RG);
            command.Parameters.AddWithValue("@DataNascimento", funcionarios.DataNascimento);
            command.Parameters.AddWithValue("@EstadoCivil", funcionarios.EstadoCivil);
            command.Parameters.AddWithValue("@EnderecoCompleto", funcionarios.EnderecoCompleto);
            command.Parameters.AddWithValue("@PIS", funcionarios.PIS);
            command.Parameters.AddWithValue("@TituloEleitor", funcionarios.TituloEleitor);
            command.Parameters.AddWithValue("@CTPS", funcionarios.CTPS);
            command.Parameters.AddWithValue("@Reservista", funcionarios.Reservista);
            command.Parameters.AddWithValue("@Cargo", funcionarios.Cargo);
            command.Parameters.AddWithValue("@SalarioBase", funcionarios.SalarioBase);
            command.Parameters.AddWithValue("@IndicadorPlanoSaude", funcionarios.IndicadorPlanoSaude);
            command.Parameters.AddWithValue("@IndicadorPlanoOdonto", funcionarios.IndicadorPlanoOdonto);
            command.Parameters.AddWithValue("@IndicadorValeTransporte", funcionarios.IndicadorValeTransporte);
            command.Parameters.AddWithValue("@IndicadorValeRefeicao", funcionarios.IndicadorValeRefeicao);
            command.Parameters.AddWithValue("@IndicadorValeAlimentacao", funcionarios.IndicadorValeAlimentacao);
            command.Parameters.AddWithValue("@Banco", funcionarios.Banco);
            command.Parameters.AddWithValue("@Agencia", funcionarios.Agencia);
            command.Parameters.AddWithValue("@Conta", funcionarios.Conta);

            command.ExecuteNonQuery();

            MessageBox.Show("Dados incluídos com sucesso!");
        }
        catch (Exception ex)
        {
            MessageBox.Show("Erro: " + ex.Message);
        }
        finally
        {
            connection.Close();
        }
```

```
1 reference
public Funcionarios read(Funcionarios funcionarios)
{
    try
    {
        string commandSql = "SELECT * FROM Funcionarios WHERE CPF = @CPF";
        SqlCommand command = new SqlCommand(commandSql, connection);

        connection.Open();

        command.Parameters.AddWithValue("@CPF", funcionarios.CPF);

        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            funcionarios.Nome = Convert.ToString(reader["Nome"]);
            funcionarios.CPF = Convert.ToString(reader["CPF"]);
            funcionarios.RG = Convert.ToString(reader["RG"]);
            funcionarios.EstadoCivil = Convert.ToString(reader["EstadoCivil"]);
            funcionarios.DataNascimento = Convert.ToString(reader["DataNascimento"]);
            funcionarios.EnderecoCompleto = Convert.ToString(reader["EnderecoCompleto"]);
            funcionarios.PIS = Convert.ToString(reader["PIS"]);
            funcionarios.TituloEleitor = Convert.ToString(reader["TituloEleitor"]);
            funcionarios.CTPS = Convert.ToString(reader["CTPS"]);
            funcionarios.Reservista = Convert.ToString(reader["Reservista"]);
            funcionarios.Cargo = Convert.ToString(reader["Cargo"]);
            funcionarios.SalarioBase = Convert.ToDecimal(reader["SalarioBase"]);
            funcionarios.IndicadorPlanoSaude = Convert.ToString(reader["IndicadorPlanoSaude"]);
            funcionarios.IndicadorPlanoOdonto = Convert.ToString(reader["IndicadorPlanoOdonto"]);
            funcionarios.IndicadorValeTransporte = Convert.ToString(reader["IndicadorValeTransporte"]);
            funcionarios.IndicadorValeRefeicao = Convert.ToString(reader["IndicadorValeRefeicao"]);
            funcionarios.IndicadorValeAlimentacao = Convert.ToString(reader["IndicadorValeAlimentacao"]);
            funcionarios.Banco = Convert.ToString(reader["Banco"]);
            funcionarios.Agencia = Convert.ToString(reader["Agencia"]);
            funcionarios.Conta = Convert.ToString(reader["Conta"]);
        }
        return funcionarios;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
        return null;
    }
    finally
    {
        connection.Close();
    }
}
```

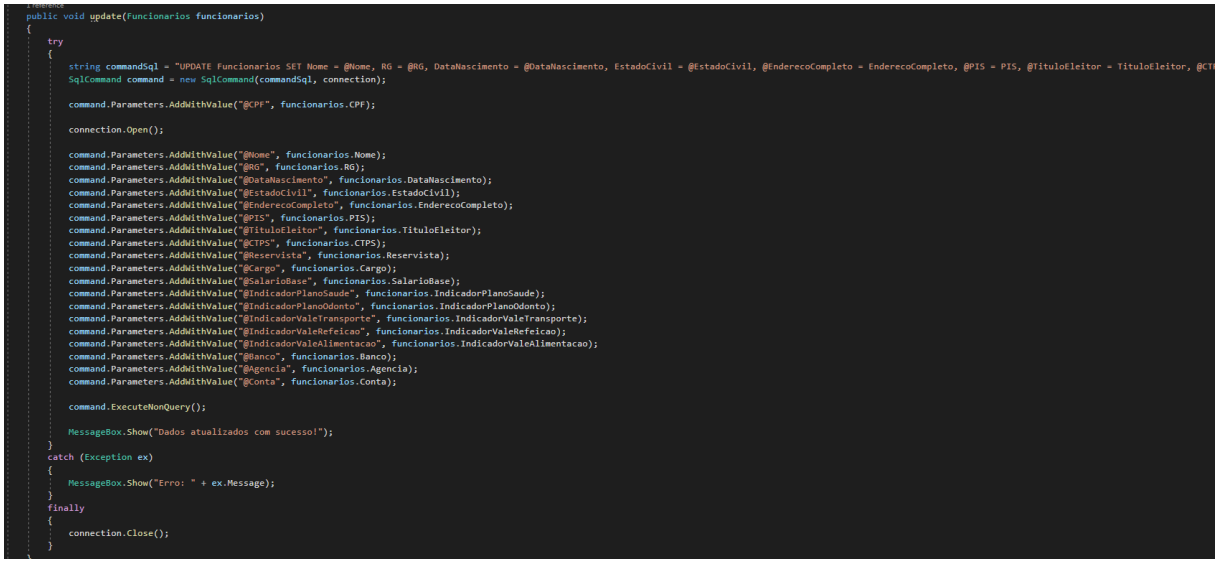
```
1 reference
public void delete(Funcionarios funcionarios)
{
    try
    {
        string commandSql = "DELETE FROM Funcionarios WHERE CPF = @CPF";
        SqlCommand command = new SqlCommand(commandSql, connection);

        connection.Open();

        command.Parameters.AddWithValue("@CPF", funcionarios.CPF);

        command.ExecuteNonQuery();

        MessageBox.Show("Funcionário excluído com sucesso.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
    }
    finally
    {
        connection.Close();
    }
}
```



```
1 reference
public void update(funcionarios funcionarios)
{
    try
    {
        string commandSql = "UPDATE Funcionarios SET Nome = @Nome, RG = @RG, DataNascimento = @DataNascimento, EstadoCivil = @EstadoCivil, @EnderecoCompleto = EnderecoCompleto, @PIS = PIS, @TituloEleitor = TituloEleitor, @CT
        SqlCommand command = new SqlCommand(commandSql, connection);

        command.Parameters.AddWithValue("@CPF", funcionarios.CPF);

        connection.Open();

        command.Parameters.AddWithValue("@Nome", funcionarios.Nome);
        command.Parameters.AddWithValue("@RG", funcionarios.RG);
        command.Parameters.AddWithValue("@DataNascimento", funcionarios.DataNascimento);
        command.Parameters.AddWithValue("@EstadoCivil", funcionarios.EstadoCivil);
        command.Parameters.AddWithValue("@EnderecoCompleto", funcionarios.EnderecoCompleto);
        command.Parameters.AddWithValue("@PIS", funcionarios.PIS);
        command.Parameters.AddWithValue("@TituloEleitor", funcionarios.TituloEleitor);
        command.Parameters.AddWithValue("@CTPS", funcionarios.CTPS);
        command.Parameters.AddWithValue("@Reservista", funcionarios.Reservista);
        command.Parameters.AddWithValue("@Cargo", funcionarios.Cargo);
        command.Parameters.AddWithValue("@SalarioBase", funcionarios.SalarioBase);
        command.Parameters.AddWithValue("@IndicadorPlanoSaude", funcionarios.IndicadorPlanoSaude);
        command.Parameters.AddWithValue("@IndicadorPlanoOdonto", funcionarios.IndicadorPlanoOdonto);
        command.Parameters.AddWithValue("@IndicadorValeTransporte", funcionarios.IndicadorValeTransporte);
        command.Parameters.AddWithValue("@IndicadorValeRefeicao", funcionarios.IndicadorValeRefeicao);
        command.Parameters.AddWithValue("@IndicadorValeAlimentacao", funcionarios.IndicadorValeAlimentacao);
        command.Parameters.AddWithValue("@Banco", funcionarios.Banco);
        command.Parameters.AddWithValue("@Agencia", funcionarios.Agencia);
        command.Parameters.AddWithValue("@Conta", funcionarios.Conta);

        command.ExecuteNonQuery();

        MessageBox.Show("Dados atualizados com sucesso!");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
    }
    finally
    {
        connection.Close();
    }
}
```

```
1 reference
public bool validarFuncionario(Funcionarios funcionarios)
{
    try
    {
        string commandSql = "SELECT * FROM Funcionarios WHERE CPF = @CPF";
        SqlCommand command = new SqlCommand(commandSql, connection);

        connection.Open();

        command.Parameters.AddWithValue("@CPF", funcionarios.CPF);

        using (SqlDataReader reader = command.ExecuteReader())
        {
            return reader.HasRows;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
        return false;
    }
    finally
    {
        connection.Close();
    }
}
```

2. Classes FolhaDePagamento

```
4 references
public partial class FolhaPagamento : Form
{
    FolhaDePagamento folhaDePagamento;
    Descontos descontos;
    Proventos proventos;
    CtrFolhaDePagamento ctrFolhaDePagamento;

    1 reference
    public FolhaPagamento()
    {
        InitializeComponent();
        folhaDePagamento = new FolhaDePagamento();
        descontos = new Descontos();
        proventos = new Proventos();
        ctrFolhaDePagamento = new CtrFolhaDePagamento();
    }

    1 reference
    private void btnSalvar_Click(object sender, EventArgs e)
    {

        if (ValidarCampos())
        {
            folhaDePagamento.CPF = txtCpf.Text;
            bool cpfExists = ctrFolhaDePagamento.validarCpfFuncionario(folhaDePagamento);

            if (cpfExists)
            {

                folhaDePagamento.CPF = txtCpf.Text;
                ctrFolhaDePagamento.readFuncionarios(descontos, folhaDePagamento);
                folhaDePagamento.MesReferencia = txtMesReferencia.Text;
                proventos.AdicionalNoturno = Convert.ToString(txtAdNoturno.Text);
                proventos.HorasExtras = Convert.ToString(txtHorasExtras.Text);
                proventos.Comissao = Convert.ToDecimal(txtComissao.Text);
                descontos.QtdeDiasFaltas = Convert.ToInt32(txtDiasFaltas.Text);
                descontos.QtdeDiasTrabalhados = Convert.ToInt32(txtDiasTrabalhados.Text);
                ctrFolhaDePagamento.createFolhaPagamento(descontos, proventos, folhaDePagamento);
                LimparDadosTxt();
            }
            else
            {
                MessageBox.Show("CPF não encontrado no banco de dados.");
            }
        }
        else
        {
            MessageBox.Show("Preencher todos os campos obrigatoriamente.");
        }
    }
```

```
1 reference
private void btnBuscar_Click(object sender, EventArgs e)
{
    if (txtCpf.MaskCompleted && txtMesReferencia.MaskCompleted)
    {
        folhaDePagamento.CPF = txtCpf.Text;
        bool cpfExists = ctrFolhaDePagamento.validarCpfFolhaPagamento(folhaDePagamento);

        if (cpfExists)
        {
            folhaDePagamento.MesReferencia = txtMesReferencia.Text;
            ctrFolhaDePagamento.readFolhaPagamento(folhaDePagamento);
            lblNomeFolha.Text = folhaDePagamento.Nome;
            lblCpfFolha.Text = folhaDePagamento.CPF;
            lblMesReferencia.Text = folhaDePagamento.MesReferencia;
            lblAdicionalNoturno.Text = Convert.ToString(folhaDePagamento.ProventoAdicionalNoturno);
            lblHorasExtras.Text = Convert.ToString(folhaDePagamento.ProventoHorasExtras);
            lblComissao.Text = Convert.ToString(folhaDePagamento.ProventoComissao);
            lblInss.Text = Convert.ToString(folhaDePagamento.DescontoINSS);
            lblIrrf.Text = Convert.ToString(folhaDePagamento.DescontoIRRF);
            lblValeTransporte.Text = Convert.ToString(folhaDePagamento.DescontoVT);
            lblValeAlimentacao.Text = Convert.ToString(folhaDePagamento.DescontoVA);
            lblValeRefeicao.Text = Convert.ToString(folhaDePagamento.DescontoVR);
            lblPlanoOdonto.Text = Convert.ToString(folhaDePagamento.DescontoPO);
            lblPlanoSaude.Text = Convert.ToString(folhaDePagamento.DescontoPS);
            lblSalarioBase.Text = Convert.ToString(folhaDePagamento.SalarioBase);
            lblSalarioBruto.Text = Convert.ToString(folhaDePagamento.SalarioBruto);
            lblSalarioLiquido.Text = Convert.ToString(folhaDePagamento.SalarioLiquido);
        }
        else
        {
            MessageBox.Show("CPF não encontrado no banco de dados.");
        }
    }
    else
        MessageBox.Show("CPF e datas inválidos");
}

1 reference
private void btnLimpar_Click(object sender, EventArgs e)
{
    LimparDadosTxt();
}
```

```
1 reference
private bool ValidarCampos()
{
    if (txtCpf.Text != "" && txtMesReferencia.Text != "" && txtAdNoturno.Text != "" && txtHorasExtras.Text != "" &&
        txtComissao.Text != "" && txtDiasTrabalhados.Text != "" && txtDiasFaltas.Text != "")
        return true;
    else return false;
}

1 reference
private void btnFechar_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Deseja realmente fechar o formulário?", "Confirmação de Fechamento", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (result == DialogResult.Yes)
    {
        this.Close();
    }
}

1 reference
private void btnLimparDados2_Click(object sender, EventArgs e)
{
    txtCpf.Text = "";
    txtMesReferencia.Text = "";
    lblNomeFolha.Text = "";
    lblMesReferencia.Text = "";
    lblCpfFolha.Text = "";
    lblAdicionalNoturno.Text = "";
    lblHorasExtras.Text = "";
    lblComissao.Text = "";
    lblInss.Text = "";
    lblIrrf.Text = "";
    lblValeTransporte.Text = "";
    lblValeAlimentacao.Text = "";
    lblValeRefeicao.Text = "";
    lblPlanoOdonto.Text = "";
    lblPlanoSaude.Text = "";
    lblSalarioBase.Text = "";
    lblSalarioBruto.Text = "";
    lblSalarioLiquido.Text = "";
    txtCpf.Focus();
}

2 references
private void LimparDadosTxt()
{
    txtCpf.Text = "";
    txtMesReferencia.Text = "";
    txtAdNoturno.Text = "";
    txtHorasExtras.Text = "";
    txtComissao.Text = "";
    txtDiasTrabalhados.Text = "";
    txtDiasFaltas.Text = "";
    txtCpf.Focus();
}
}
```

```
namespace PIM_IV.Models
{
    27 references
    class FolhaDePagamento
    {
        4 references
        public string Nome { get; set; }
        10 references
        public string CPF { get; set; }
        6 references
        public string MesReferencia { get; set; }
        8 references
        public decimal ProventoHorasExtras { get; set; }
        8 references
        public decimal ProventoComissao { get; set; }
        8 references
        public decimal ProventoAdicionalNoturno { get; set; }
        20 references
        public decimal DescontoINSS { get; set; }
        21 references
        public decimal DescontoIRRF { get; set; }
        13 references
        public decimal DescontoVT { get; set; }
        9 references
        public decimal DescontoVA { get; set; }
        13 references
        public decimal DescontoVR { get; set; }
        9 references
        public decimal DescontoPO { get; set; }
        9 references
        public decimal DescontoPS { get; set; }
        9 references
        public decimal SalarioBase { get; set; }
        23 references
        public decimal SalarioBruto { get; set; }
        7 references
        public decimal SalarioLiquido { get; set; }

    }
}
```

```
namespace PIM_IV.Controllers
{
    3 references
    class CtrFolhaDePagamento
    {
        Dao_FolhaPagamento dao_FolhaPagamento;
        CtrProventos ctrProventos;
        CtrDescontos ctrDescontos;

        1 reference
        public CtrFolhaDePagamento()
        {
            dao_FolhaPagamento = new Dao_FolhaPagamento();
            ctrProventos = new CtrProventos();
            ctrDescontos = new CtrDescontos();
        }

        1 reference
        public void readFuncionarios(Descontos descontos, FolhaDePagamento folhaDePagamento)
        {
            dao_FolhaPagamento.readFuncionarios(descontos,folhaDePagamento);
        }

        1 reference
        public void createFolhaPagamento(Descontos descontos, Proventos proventos, FolhaDePagamento folhaDePagamento)
        {
            ctrProventos.Proventos(proventos, folhaDePagamento);
            ctrDescontos.Descontos(descontos, folhaDePagamento);
            dao_FolhaPagamento.create(folhaDePagamento);
        }

        1 reference
        public void readFolhaPagamento(FolhaDePagamento folhaDePagamento)
        {
            dao_FolhaPagamento.read(folhaDePagamento);
        }

        1 reference
        public bool validarCpfFuncionario(FolhaDePagamento folhaDePagamento)
        {
            return dao_FolhaPagamento.validarCpfFuncionario(folhaDePagamento);
        }

        1 reference
        public bool validarCpfFolhaPagamento(FolhaDePagamento folhaDePagamento)
        {
            return dao_FolhaPagamento.validarCpfFolhaPagamento(folhaDePagamento);
        }
    }
}
```

```
namespace PIM_IV.DAO
{
    2 references
    class Dao_FolhaPagamento
    {
        SqlConnection connection = new SqlConnection(@"Data Source=VIEIRMC-W11ASUS\SQLSERVER2022;Initial Catalog=PIM_IV;Integrated Security=True;");

        1 reference
        public Descontos readFuncionarios(Descontos descontos, FolhaDePagamento folhaDePagamento)
        {
            try
            {
                string commandSql = "SELECT * FROM Funcionarios WHERE CPF = @CPF";
                SqlCommand command = new SqlCommand(commandSql, connection);

                connection.Open();

                command.Parameters.AddWithValue("@CPF", folhaDePagamento.CPF);

                SqlDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    folhaDePagamento.Nome = Convert.ToString(reader["Nome"]);
                    folhaDePagamento.SalarioBase = Convert.ToDecimal(reader["SalarioBase"]);
                    descontos.IndicadorPlanoSaude = Convert.ToString(reader["IndicadorPlanoSaude"]);
                    descontos.IndicadorPlanoOdonto = Convert.ToString(reader["IndicadorPlanoOdonto"]);
                    descontos.IndicadorValeTransporte = Convert.ToString(reader["IndicadorValeTransporte"]);
                    descontos.IndicadorValeRefeicao = Convert.ToString(reader["IndicadorValeRefeicao"]);
                    descontos.IndicadorValeAlimentacao = Convert.ToString(reader["IndicadorValeAlimentacao"]);
                }
                return descontos;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Erro: " + ex.Message);
                return null;
            }
            finally
            {
                connection.Close();
            }
        }
```

```
1 reference
public void create(FolhaDePagamento folhaDePagamento)
{
    try
    {
        string commandSql = "INSERT INTO FolhaDePagamento(Nome,CPF,MesReferencia,ProventoAdicionalNoturno,ProventoHorasExtras,ProventoComissao,DescontoINSS,DescontoIRRF,DescontoVT,DescontoVA,DescontoVR,DescontoPS,DescontoPO,
        SqlCommand command = new SqlCommand(commandSql, connection);

        connection.Open();

        command.Parameters.AddWithValue("@Nome", folhaDePagamento.Nome);
        command.Parameters.AddWithValue("@CPF", folhaDePagamento.CPF);
        command.Parameters.AddWithValue("@MesReferencia", folhaDePagamento.MesReferencia);
        command.Parameters.AddWithValue("@ProventoAdicionalNoturno", folhaDePagamento.ProventoAdicionalNoturno);
        command.Parameters.AddWithValue("@ProventoHorasExtras", folhaDePagamento.ProventoHorasExtras);
        command.Parameters.AddWithValue("@ProventoComissao", folhaDePagamento.ProventoComissao);
        command.Parameters.AddWithValue("@DescontoINSS", folhaDePagamento.DescontoINSS);
        command.Parameters.AddWithValue("@DescontoIRRF", folhaDePagamento.DescontoIRRF);
        command.Parameters.AddWithValue("@DescontoVT", folhaDePagamento.DescontoVT);
        command.Parameters.AddWithValue("@DescontoVA", folhaDePagamento.DescontoVA);
        command.Parameters.AddWithValue("@DescontoVR", folhaDePagamento.DescontoVR);
        command.Parameters.AddWithValue("@DescontoPS", folhaDePagamento.DescontoPS);
        command.Parameters.AddWithValue("@DescontoPO", folhaDePagamento.DescontoPO);
        command.Parameters.AddWithValue("@SalarioBase", folhaDePagamento.SalarioBase);
        command.Parameters.AddWithValue("@SalarioBruto", folhaDePagamento.SalarioBruto);
        command.Parameters.AddWithValue("@SalarioLiquido", folhaDePagamento.SalarioLiquido);

        command.ExecuteNonQuery();

        MessageBox.Show("Registros incluídos com sucesso!");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
    }
    finally
    {
        connection.Close();
    }
}
```

```
public FolhaDePagamento read(FolhaDePagamento folhaDePagamento)
{
    try
    {
        string commandSql = "SELECT * FROM FolhaDePagamento WHERE CPF = @CPF AND MesReferencia = @MesReferencia";
        SqlCommand command = new SqlCommand(commandSql, connection);

        connection.Open();

        command.Parameters.AddWithValue("@CPF", folhaDePagamento.CPF);
        command.Parameters.AddWithValue("@MesReferencia", folhaDePagamento.MesReferencia);

        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            folhaDePagamento.Nome = Convert.ToString(reader["Nome"]);
            folhaDePagamento.CPF = Convert.ToString(reader["CPF"]);
            folhaDePagamento.MesReferencia = Convert.ToString(reader["MesReferencia"]);
            folhaDePagamento.ProventoAdicionalNoturno = Convert.ToDecimal(reader["ProventoAdicionalNoturno"]);
            folhaDePagamento.ProventoHorasExtras = Convert.ToDecimal(reader["ProventoHorasExtras"]);
            folhaDePagamento.ProventoComissao = Convert.ToDecimal(reader["ProventoComissao"]);
            folhaDePagamento.DescontoINSS = Convert.ToDecimal(reader["DescontoINSS"]);
            folhaDePagamento.DescontoIRRF = Convert.ToDecimal(reader["DescontoIRRF"]);
            folhaDePagamento.DescontoVT = Convert.ToDecimal(reader["DescontoVT"]);
            folhaDePagamento.DescontoVA = Convert.ToDecimal(reader["DescontoVA"]);
            folhaDePagamento.DescontoVR = Convert.ToDecimal(reader["DescontoVR"]);
            folhaDePagamento.DescontoPO = Convert.ToDecimal(reader["DescontoPO"]);
            folhaDePagamento.DescontoPS = Convert.ToDecimal(reader["DescontoPS"]);
            folhaDePagamento.SalarioBase = Convert.ToDecimal(reader["SalarioBase"]);
            folhaDePagamento.SalarioBruto = Convert.ToDecimal(reader["SalarioBruto"]);
            folhaDePagamento.SalarioLiquido = Convert.ToDecimal(reader["SalarioLiquido"]);
        }
        return folhaDePagamento;
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
        return null;
    }
    finally
    {
        connection.Close();
    }
}
```

```
1 reference
public bool validarCpfFuncionario(FolhaDePagamento folhaDePagamento)
{
    try
    {
        string commandSql = "SELECT * FROM Funcionarios WHERE CPF = @CPF";
        SqlCommand command = new SqlCommand(commandSql, connection);

        connection.Open();

        command.Parameters.AddWithValue("@CPF", folhaDePagamento.CPF);

        using (SqlDataReader reader = command.ExecuteReader())
        {
            return reader.HasRows;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
        return false;
    }
    finally
    {
        connection.Close();
    }
}

1 reference
public bool validarCpfFolhaPagamento(FolhaDePagamento folhaDePagamento)
{
    try
    {
        string commandSql = "SELECT * FROM FolhaDePagamento WHERE CPF = @CPF";
        SqlCommand command = new SqlCommand(commandSql, connection);

        connection.Open();

        command.Parameters.AddWithValue("@CPF", folhaDePagamento.CPF);

        using (SqlDataReader reader = command.ExecuteReader())
        {
            return reader.HasRows;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro: " + ex.Message);
        return false;
    }
    finally
    {
        connection.Close();
    }
}
```

3. Classes Descontos

```
namespace PIM_IV.Models
{
    12 references
    class Descontos
    {
        5 references
        public int QtdeDiasTrabalhados { get; set; }
        5 references
        public int QtdeDiasFaltas { get; set; }
        2 references
        public string IndicadorPlanoSaude { get; set; }
        2 references
        public string IndicadorPlanoOdonto { get; set; }
        2 references
        public string IndicadorValeTransporte { get; set; }
        2 references
        public string IndicadorValeRefeicao { get; set; }
        2 references
        public string IndicadorValeAlimentacao { get; set; }
    }
}
```

```
namespace PIM_IV.Controllers
{
    2 references
    class CtrDescontos
    {
        1 reference
        public decimal CalcularINSS(FolhaDePagamento folhaDePagamento)
        {
            if (folhaDePagamento.SalarioBruto <= 1320)
            {
                folhaDePagamento.DescontoINSS = folhaDePagamento.SalarioBruto * 0.075m;
                folhaDePagamento.DescontoINSS = Math.Round(folhaDePagamento.DescontoINSS, 2);
                return folhaDePagamento.DescontoINSS;
            }
            else if (folhaDePagamento.SalarioBruto <= 2571.29m)
            {
                folhaDePagamento.DescontoINSS = folhaDePagamento.SalarioBruto * 0.09m;
                folhaDePagamento.DescontoINSS = Math.Round(folhaDePagamento.DescontoINSS, 2);
                return folhaDePagamento.DescontoINSS;
            }
            else if (folhaDePagamento.SalarioBruto <= 3856.94m)
            {
                folhaDePagamento.DescontoINSS = folhaDePagamento.SalarioBruto * 0.12m;
                folhaDePagamento.DescontoINSS = Math.Round(folhaDePagamento.DescontoINSS, 2);
                return folhaDePagamento.DescontoINSS;
            }
            else
            {
                folhaDePagamento.DescontoINSS = folhaDePagamento.SalarioBruto * 0.14m;
                folhaDePagamento.DescontoINSS = Math.Round(folhaDePagamento.DescontoINSS, 2);
                return folhaDePagamento.DescontoINSS;
            }
        }
```

```
1 reference
public decimal CalcularIRRF(FolhaDePagamento folhaDePagamento)
{
    if (folhaDePagamento.SalarioBruto <= 2112)
        return folhaDePagamento.DescontoIRRF = 0;
    else if (folhaDePagamento.SalarioBruto <= 2826.65m)
    {
        folhaDePagamento.DescontoIRRF = (folhaDePagamento.SalarioBruto * 0.075m) - 142.80m;
        folhaDePagamento.DescontoIRRF = Math.Round(folhaDePagamento.DescontoIRRF, 2);
        return folhaDePagamento.DescontoIRRF;
    }
    else if (folhaDePagamento.SalarioBruto <= 3751.05m)
    {
        folhaDePagamento.DescontoIRRF = (folhaDePagamento.SalarioBruto * 0.15m) - 354.80m;
        folhaDePagamento.DescontoIRRF = Math.Round(folhaDePagamento.DescontoIRRF, 2);
        return folhaDePagamento.DescontoIRRF;
    }
    else if (folhaDePagamento.SalarioBruto <= 4664.68m)
    {
        folhaDePagamento.DescontoIRRF = (folhaDePagamento.SalarioBruto * 0.225m) - 636.13m;
        folhaDePagamento.DescontoIRRF = Math.Round(folhaDePagamento.DescontoIRRF, 2);
        return folhaDePagamento.DescontoIRRF;
    }
    else
    {
        folhaDePagamento.DescontoIRRF = (folhaDePagamento.SalarioBruto * 0.275m) - 869.36m;
        folhaDePagamento.DescontoIRRF = Math.Round(folhaDePagamento.DescontoIRRF, 2);
        return folhaDePagamento.DescontoIRRF;
    }
}

1 reference
public decimal CalcularPlanoSaude(Descontos descontos, FolhaDePagamento folhaDePagamento)
{
    if (descontos.IndicadorPlanoSaude == "Sim")
    {
        folhaDePagamento.DescontoPS = folhaDePagamento.SalarioBase * 0.03m;
        folhaDePagamento.DescontoPS = Math.Round(folhaDePagamento.DescontoPS, 2);
        return folhaDePagamento.DescontoPS;
    }
    else
        return folhaDePagamento.DescontoPS = 0;
}
```

```
1 reference
public decimal CalcularPlanoOdontologico(Descontos descontos, FolhaDePagamento folhaDePagamento)
{
    if (descontos.IndicadorPlanoOdonto == "Sim")
    {
        folhaDePagamento.DescontoPO = folhaDePagamento.SalarioBase * 0.02m;
        folhaDePagamento.DescontoPO = Math.Round(folhaDePagamento.DescontoPO, 2);
        return folhaDePagamento.DescontoPO;
    }
    else
        return folhaDePagamento.DescontoPO = 0;
}

1 reference
public decimal CalcularValeTransporte(Descontos descontos, FolhaDePagamento folhaDePagamento)
{
    if (descontos.IndicadorValeTransporte == "Sim")
    {
        if (descontos.QtdeDiasFaltas == 0)
        {
            //Passagem Campinas ida e volta: 10,80
            decimal valeTransporteoPorDia = 10.80m;
            folhaDePagamento.DescontoVT = (valeTransporteoPorDia * descontos.QtdeDiasTrabalhados) * 0.06m;
            folhaDePagamento.DescontoVT = Math.Round(folhaDePagamento.DescontoVT, 2);
            return folhaDePagamento.DescontoVT;
        }
        else
        {
            decimal valeTransporteoPorDia = 10.80m;
            folhaDePagamento.DescontoVT = (valeTransporteoPorDia * (descontos.QtdeDiasTrabalhados - descontos.QtdeDiasFaltas)) * 0.06m;
            folhaDePagamento.DescontoVT = Math.Round(folhaDePagamento.DescontoVT, 2);
            return folhaDePagamento.DescontoVT;
        }
    }
    else
        return folhaDePagamento.DescontoVT = 0;
}
```

```
1 reference
public decimal CalcularValeAlimentacao(Descontos descontos, FolhaDePagamento folhaDePagamento)
{
    if (descontos.IndicadorValeAlimentacao == "Sim")
    {
        decimal valeAlimentacao = 400;
        folhaDePagamento.DescontoVA = valeAlimentacao * 0.10m;
        folhaDePagamento.DescontoVA = Math.Round(folhaDePagamento.DescontoVA, 2);
        return folhaDePagamento.DescontoVA;
    }
    else
        return folhaDePagamento.DescontoVA = 0;
}

1 reference
public decimal CalcularValeRefeicao(Descontos descontos, FolhaDePagamento folhaDePagamento)
{
    if (descontos.IndicadorValeRefeicao == "Sim")
    {
        if (descontos.QtdeDiasFaltas == 0)
        {
            decimal valeRefeicaoPorDia = 16;
            folhaDePagamento.DescontoVR = (valeRefeicaoPorDia * descontos.QtdeDiasTrabalhados) * 0.10m;
            folhaDePagamento.DescontoVR = Math.Round(folhaDePagamento.DescontoVR, 2);
            return folhaDePagamento.DescontoVR;
        }
        else
        {
            decimal valeRefeicaoPorDia = 10.80m;
            folhaDePagamento.DescontoVR = (valeRefeicaoPorDia * (descontos.QtdeDiasTrabalhados - descontos.QtdeDiasFaltas)) * 0.10m;
            folhaDePagamento.DescontoVR = Math.Round(folhaDePagamento.DescontoVR, 2);
            return folhaDePagamento.DescontoVR;
        }
    }
    else
        return folhaDePagamento.DescontoVR = 0;
}
```

```
1 reference
public decimal SalarioLiquido(FolhaDePagamento folhaDePagamento)
{
    var calculo = folhaDePagamento.DescontoINSS + folhaDePagamento.DescontoIRRF + folhaDePagamento.DescontoPO + folhaDePagamento.DescontoPS
        + folhaDePagamento.DescontoVA + folhaDePagamento.DescontoVR + folhaDePagamento.DescontoVT;
    folhaDePagamento.SalarioLiquido = folhaDePagamento.SalarioBruto - calculo;
    folhaDePagamento.SalarioLiquido = Math.Round(folhaDePagamento.SalarioLiquido, 2);
    return folhaDePagamento.SalarioLiquido;
}

1 reference
public void Descontos(Descontos descontos, FolhaDePagamento folhaDePagamento)
{
    CalcularINSS(folhaDePagamento);
    CalcularIRRF(folhaDePagamento);
    CalcularPlanoSaude(descontos, folhaDePagamento);
    CalcularPlanoOdontologico(descontos, folhaDePagamento);
    CalcularValeTransporte(descontos, folhaDePagamento);
    CalcularValeAlimentacao(descontos, folhaDePagamento);
    CalcularValeRefeicao(descontos, folhaDePagamento);
    SalarioLiquido(folhaDePagamento);
}
```

4. Classes Proventos

```
namespace PIM_IV.Models
{
    7 references
    class Proventos
    {
        2 references
        public string AdicionalNoturno { get; set; }
        2 references
        public string HorasExtras { get; set; }
        2 references
        public decimal Comissao { get; set; }
    }
}
```

```
namespace PIM_IV.Controllers
{
    2 references
    class CtrProventos
    {
        1 reference
        public decimal CalcularHorasExtras(Proventos proventos, FolhaDePagamento folhaDePagamento)
        {
            TimeSpan.TryParse(proventos.HorasExtras, out TimeSpan horasExtras);

            double minutosExtras = horasExtras.TotalMinutes;
            var valorPorMinuto = folhaDePagamento.SalarioBase / (220 * 60);

            folhaDePagamento.ProventoHorasExtras = valorPorMinuto * (decimal)minutosExtras;
            folhaDePagamento.ProventoHorasExtras = Math.Round(folhaDePagamento.ProventoHorasExtras, 2);

            return folhaDePagamento.ProventoHorasExtras;
        }

        1 reference
        public decimal CalcularComissao(Proventos proventos, FolhaDePagamento folhaDePagamento)
        {
            folhaDePagamento.ProventoComissao = proventos.Comissao * 0.03m;
            folhaDePagamento.ProventoComissao = Math.Round(folhaDePagamento.ProventoComissao, 2);
            return folhaDePagamento.ProventoComissao;
        }
```

```
1 reference
public decimal CalcularAdicionalNoturno(Proventos proventos, FolhaDePagamento folhaDePagamento)
{
    TimeSpan.TryParse(proventos.AdicionalNoturno, out TimeSpan adicionalNoturno);

    double minutosAdicional = adicionalNoturno.TotalMinutes;
    var valorPorMinuto = folhaDePagamento.SalarioBase / (220 * 60);

    folhaDePagamento.ProventoAdicionalNoturno = (decimal)minutosAdicional * (valorPorMinuto * 1.2m);
    folhaDePagamento.ProventoAdicionalNoturno = Math.Round(folhaDePagamento.ProventoAdicionalNoturno, 2);

    return folhaDePagamento.ProventoAdicionalNoturno;
}

1 reference
public decimal SalarioBruto(FolhaDePagamento folhaDePagamento)
{
    var calculo = folhaDePagamento.ProventoAdicionalNoturno + folhaDePagamento.ProventoHorasExtras + folhaDePagamento.ProventoComissao;
    folhaDePagamento.SalarioBruto = folhaDePagamento.SalarioBase + calculo;
    folhaDePagamento.SalarioBruto = Math.Round(folhaDePagamento.SalarioBruto, 2);
    return folhaDePagamento.SalarioBruto;
}

1 reference
public void Proventos(Proventos proventos, FolhaDePagamento folhaDePagamento)
{
    CalcularHorasExtras(proventos, folhaDePagamento);
    CalcularComissao(proventos, folhaDePagamento);
    CalcularAdicionalNoturno(proventos, folhaDePagamento);
    SalarioBruto(folhaDePagamento);
}
```

5. Classe TelaInicial

```
3 references
public partial class TelaInicial : Form
{
    1 reference
    public TelaInicial()
    {
        InitializeComponent();
    }

    1 reference
    private void btnCadastrarFuncionarios_Click(object sender, EventArgs e)
    {
        CadastrarFuncionarios cadastrarFuncionarios = new CadastrarFuncionarios();
        cadastrarFuncionarios.ShowDialog();
    }

    1 reference
    private void btnFolhaPagamento_Click(object sender, EventArgs e)
    {
        FolhaPagamento folhaPagamento = new FolhaPagamento();
        folhaPagamento.ShowDialog();
    }

    1 reference
    private void btnFechar_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Deseja realmente fechar o formulário?", "Confirmação de Fechamento", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

        if (result == DialogResult.Yes)
        {
            this.Close();
        }
    }
}
```