
Especificação

do

Sistema Contábil Ribanceira Recursos Humanos

Versão 1.7

Alessandra Harumi Iriguti	ra88741
Christian Takashi Nakata	ra90558
Kevin Levrone Rodrigues Machado Silva	ra89275
Pedro Henrique Torres Peres Garozi *	ra90552

Professor (a): Donizete Carlos Bruzarosco

Disciplina: Implementação de Sistemas de Software

Sumário

1	INTRODUÇÃO	5
1.1	OBJETIVO DO DOCUMENTO.....	5
1.2	ESCOPO DO PRODUTO	5
1.3	PÚBLICO-ALVO	5
1.4	DEFINIÇÕES, ACRÔNIMOS E ABREVIACÕES	5
1.5	CONVENÇÕES.....	6
1.6	REFERÊNCIAS.....	6
2	VISÃO GERAL.....	7
2.1	PERSPECTIVA DO PRODUTO	7
2.2	FUNCIONALIDADE DO PRODUTO.....	7
2.3	USUÁRIOS.....	7
2.4	AMBIENTE OPERACIONAL.....	7
2.5	RESTRIÇÕES DE PROJETO E IMPLEMENTAÇÃO	7
2.6	DOCUMENTAÇÃO DO USUÁRIO.....	7
2.7	SUPosições e DEPENDÊNCIAS	8
3	ESPECIFICAÇÃO DAS INTERFACES EXTERNAS	9
3.1	REQUISITOS DE INTERFACE EXTERNA.....	9
4	REQUISITOS FUNCIONAIS	10
4.1	RF001 MANTER EMPRESAS.....	10
4.2	RF002 MANTER FUNCIONÁRIOS	10
4.3	RF003 MANTER CONTRATO DE TRABALHO	10
4.4	RF004 GERAR FOLHA DE PAGAMENTO	11
4.5	RF005 MANTER CONVENÇÃO COLETIVA DOS SINDICATOS.....	11
4.6	RF006 MANTER SINDICATOS	12
4.7	RF007 ALERTAR SOBRE VENCIMENTOS	12
4.8	RF008 MANTER OCORRÊNCIAS	12
4.9	RF009 CALCULAR SALÁRIO	13
4.10	RF010 REALIZAR TAREFAS DE RESCISÃO	13
4.11	RF011 REALIZAR TAREFAS DE ADMISSÃO	14
4.12	RF012 CALCULAR FÉRIAS	14
4.13	RF013 CALCULAR ABONO PECUNIÁRIO.....	14
4.14	RF014 MANTER AVISO PRÉVIO	15
5	REQUISITOS NÃO-FUNCIONAIS	16
5.1	REQUISITOS DE DESEMPENHO	16
5.2	REQUISITOS DE SEGURANÇA	16
5.3	ATRIBUTOS DE QUALIDADE DO SOFTWARE	16
6	MODELO DE CASOS DE USO	17
6.1	DESCRÍÇÃO DOS CASOS DE USO PRINCIPAIS.....	18
6.2	TABELA DE CONCEITOS E CONSULTAS	21
7	DIAGRAMA DE PACOTES	27
8	DIAGRAMA DE CLASSES	30
9	DIAGRAMA DE COMUNICAÇÃO.....	36

10 ARQUITETURA DE SOFTWARE.....	49
10.1 ESTILOS ARQUITETURAIS.....	49
10.2 PADRÓES DE PROJETO.....	50
10.3 <i>FRAMEWORKS</i>	50
10.4 ALTERAÇÕES NO PROJETO	51
11 DIAGRAMAS DE SEQUÊNCIA.....	52
12 DIAGRAMAS DE ESTADOS.....	60
13 DIAGRAMA DE ATIVIDADES.....	64
14 PLANEJAMENTO DAS SPRINTS.....	72
15 DIAGRAMA DE CLASSES MVC	73
16 <i>FRAMEWORK</i>.....	82
17 PADRÓES DE PROJETO	83
18 TESTES.....	84
19 TELAS	92

Revisões

Versão	Autores	Descrição da Versão	Data
1.0	Alessandra Iriguti, Christian Nakata, Pedro Garozi, Kevin Rodrigues.	Versão inicial do software, com a inserção dos requisitos mais importantes que foram demandados pelo cliente.	01/08/2015
1.1	Alessandra Iriguti, Christian Nakata, Pedro Garozi, Kevin Rodrigues.	Modificações nos Requisitos Funcionais, inserção de um novo Requisito Funcional	06/09/2015
1.2	Alessandra Iriguti, Christian Nakata, Pedro Garozi	Workflow da Captura de Requisitos	06/12/2015
1.3	Alessandra Iriguti, Christian Nakata, Pedro Garozi	Workflow de Análise	15/02/2016
1.4	Alessandra Iriguti, Christian Nakata, Pedro Garozi, Kevin Rodrigues.	Inclusão das classes <i>boundary</i> e <i>control</i> no diagrama de classes; inclusão dos métodos no diagrama de classes; alterações nos diagramas de comunicação; definição de estilos arquiteturais, padrões de projeto e <i>frameworks</i> ; alterações das telas.	15/06/2016
1.5	Alessandra Iriguti, Christian Nakata, Pedro Garozi, Kevin Rodrigues.	Correções nos diagramas de comunicação; inclusão dos diagramas de sequência, de estados e de atividades; melhora das telas do programa em questão.	08/08/2016
1.6	Alessandra Iriguti	Revisão dos requisitos funcionais	19/08/2016
1.7	Alessandra Iriguti	Alterações de projeto (item 10.4, página 49)	08/11/2016

1 Introdução

Pretende-se desenvolver um sistema contábil voltado a parte do setor de Recursos Humanos - RH. Em geral, um sistema exerce as funções de: Autônomos; Pro-Labore; Tomadores de Serviços; Sindicatos; Cadastro de Funcionários; Memória de Cálculo; Alerta de Vencimentos e Histórico Funcional. Por exemplo, fazendo-se o controle de pessoal, desde a execução da folha de pagamento até tratamento completo para férias, 13º salário, rescisões, vale transporte, entre outros. Centralizando, ainda, as informações e controles sindicais, conforme as convenções coletivas de cada sindicato.

1.1 Objetivo do Documento

Feita a elicitação de requisitos e sua seleção por meio de um estudo de viabilidade, é necessário padronizar os requisitos. A fim de se organizar os requisitos e suas relações. Caso ocorra alguma alteração em algum requisito, facilita observar como tal mudança impactará nos outros requisitos e no próprio sistema.

O produto de software a ser desenvolvido é um Sistema Contábil para os Recursos Humanos. Este, possui inúmeros requisitos funcionais. Tais requisitos possuem vários relacionamentos dentro do sistema. Os procedimentos do produto serão realizados, neste caso, por apenas um contador, ou seja, o usuário final é o contador autônomo que possui um escritório de pequeno porte.

1.2 Escopo do Produto

O produto deve satisfazer e não exceder as funções de um Sistema Contábil para o setor de Recursos Humanos (parte dele). Após o cadastro das empresas e dos funcionários, o sistema deverá realizar desde cálculo de honorários até alertas de vencimentos. O cliente gostaria que houvesse uma atualização automática da parte trabalhista, principalmente, da parte de sindicatos, pois, para realizá-los, o contador tem que imprimir o sindicato anterior e alterar manualmente no sistema. Além disso, o cliente gostaria que houvesse um alerta ao abrir o sistema ou uma tela após a inicial, para: (a) vencimento de férias; (b) vencimento da vigência de uma convenção coletiva de trabalho - sindicatos; (c) vencimento de contrato de experiência. Gostaria, ainda, que houvesse uma função para estar entrando em cada funcionário cadastrado e verificar tais vencimentos.

1.3 Público-Alvo

Os possíveis leitores do documento são as pessoas relacionadas ao desenvolvimento do produto. Dentre eles estão os engenheiros, os programadores e, inclusive, o usuário final. A fim de que após a observação feita por todas as partes, estes façam uma análise e uma aprovação do documento (ou uma alteração para uma nova análise).

1.4 Definições, Acrônimos e Abreviações

- RH: Recursos Humanos;
- SEFIP: É uma declaração que deve ser enviada ao governo com informações da Folha de Pagamento e da quantia de FGTS e INSS a ser pago pela empresa, com as informações de seus empregados;

- GRFC: (Guia de Recolhimento Rescisório do FGTS e da Contribuição Social) É o documento destinado ao recolhimento para o FGTS e da Contribuição Social, nos casos de dispensa do empregado sem justa causa, inclusive a indireta, por culpa recíproca, por força maior e na rescisão do contrato firmado nos termos da Lei 9.601/98;
- FGTS: É a sigla de Fundo de Garantia por Tempo de Serviço. É um depósito mensal de 8% do salário do empregado, que o empregador é obrigado a depositar em uma conta bancária no nome do empregado que deve ser aberta na Caixa Econômica Federal;
- INSS: É a sigla de Instituto Nacional do Seguro Social. Tal órgão é responsável por recolher contribuições das pessoas, pagas durante o período de trabalhadoras, a fim de realizar pagamentos de aposentadorias, auxílio doença, pensão por morte, auxílio-acidente, entre outros benefícios sob lei. As regras das contribuições variam de acordo com o contribuinte. Os INSSs são descontados na Folha de Pagamento para funcionários e os valores variam de 8 a 11% do salário;
- Folha de Pagamento: é o nome dado a uma lista, geralmente, mensal da quantia paga aos trabalhadores de uma empresa;
- CAT: É a sigla de Comunicação de Acidente de Trabalho. Serve para registrar acidentes de trabalhos e doenças ocupacionais por parte do empregador, podendo haver ou não seu afastamento do ambiente de trabalho;
- MANAD: É a sigla para Manual Normativo de Arquivos Digitais. É um modelo padrão com a lei vigente de escrituração de livros ou produção de documentos de natureza contábil, fiscal, trabalhista e previdenciária. O arquivo deve ser apresentado sob intimação de um Auditor Fiscal da Secretaria da Receita Federal;
- RAIS: Sigla de Relação Anual de Informações Sociais. É um instrumento de coleta de dados trabalhistas. A fim de se elaborar dados estatísticos do trabalho e de disponibilidade do mercado de trabalho. Além disso, assiste às necessidades de controle de atividades trabalhistas do país;
- DIRF: Declaração do Imposto de Renda Retido na Fonte é uma obrigação tributária acessória que deve ser paga por todas as pessoas jurídicas;
- CAGED: Cadastro Geral de Empregados e Desempregados é um registro administrativo (instituído pela lei nº 4923, dezembro de 1965) com o objetivo de acompanhar os processos de admissão e demissão de empregados e dar assistência aos desempregados. Além disso, subsidia programas do Ministério do Trabalho, particularmente do Seguro Desemprego;
- PIS/PASEP: PASEP é a sigla de Programa de Integração Social e do Programa de Formação do Patrimônio do Servidor Público. O PASEP é uma contribuição social devida pelas empresas. O PIS/PASEP é um número cadastrado no cartão de CNPJ ou no documento de cadastro do trabalhador. É um programa de complementação de renda do governo. Tem como objetivo financiar o pagamento de seguro-desemprego, abono e participação na receita dos órgãos e entidades para os trabalhadores. PIS/PASEP é também uma forma de segurança para o FGTS.;
- CNPJ: Cadastro Nacional da Pessoa Jurídica. É um cadastro onde todas as pessoas jurídicas e as equiparadas (pessoas físicas que exploram em nome individual atividades com intuito de lucro) são obrigadas a se inscrever antes de iniciar as suas atividades.

1.5 Convenções

Não houve uso de convenções para nenhum requisito nem documento.

1.6 Referências

Não houve uso de citações.

2 Visão Geral

2.1 Perspectiva do Produto

O sistema foi desenvolvido pela ocasião de ajudar o contador em suas tarefas diárias, cujo mesmo interage com o ambiente do contador armazenando os dados dos clientes e gerando relatórios conforme a necessidade da situação.

2.2 Funcionalidade do Produto

- 1) Cadastro de empresas;
- 2) Cadastro de funcionários das empresas;
- 3) Cadastro dos sindicatos;
- 4) Gerar contrato de trabalho;
- 5) Calcular salários, férias, abono pecuniário e impostos;
- 6) Gerar folha de pagamento;
- 7) Tarefas de (des)vinculação do funcionário com a empresa;
- 8) Alertas de vencimentos.

2.3 Usuários

O único usuário do sistema será o Contador.

2.4 Ambiente Operacional

Computador com o sistema operacional Linux.

2.5 Restrições de Projeto e Implementação

O sistema deve rodar em microcomputadores da linha IBM PC que possuam no mínimo um processador de 1 GHZ, memória RAM mínima de 512 Megabites e que execute em plataforma *Linux*.

2.6 Documentação do Usuário

O sistema de RH possui orientações por suporte técnico e um manual online para consulta.

2.7 Suposições e Dependências

Não se aplicam.

3 Especificação das Interfaces Externas

3.1 Requisitos de Interface Externa

3.1.1 Interfaces do Usuário

Será entregue em anexo ao longo do documento

3.1.2 Interfaces de Hardware

Não há produtos externos que se conectam com o sistema.

3.1.3 Interfaces de Software

O produto não possui nenhuma conexão com demais softwares.

3.1.4 Interfaces de Comunicação

O sistema abordado gera relatórios nos seguintes formatos: EBS, .RTF, .SLK.

Ademais, pela função Cordilheira Escritório Virtual - CEV -, é possível enviar os relatórios gerados, pelo sistema, via e-mail.

4 Requisitos Funcionais

4.1 RF001 Manter empresas

Descrição: Inserção, busca, alteração e exclusão de empresas no sistema.

Prioridade:	<input checked="" type="checkbox"/> Essencial	Importante	Desejável
Ator(es):	<i>Contador.</i>		
Requisitos associados:	<i>RF002; RF003; RF004; RF006; RF010; RF011.</i>		
Objetivos:	<i>Persistência dos dados das empresas</i>		
Dados de entrada:	<i>Nome da empresa, nome fantasia, CNPJ, inscrição estadual, inscrição municipal, tipo da empresa, endereço, telefone, e-mail, regime de tributação</i>		
Operações básicas:	<i>Não tem.</i>		
Dados de saída:	<i>Mensagens de aviso sobre o sucesso (ou não) de CRUD</i>		
Responsável	<i>Pedro *</i>		

4.2 RF002 Manter funcionários

Descrição: Inserção, busca, alteração e exclusão dos funcionários das empresas.

Prioridade:	<input checked="" type="checkbox"/> Essencial	Importante	Desejável
Ator(es):	<i>Contador.</i>		
Requisitos associados:	<i>RF001, RF003; RF004; RF008; RF009; RF010; RF011; RF012; RF013; RF014</i>		
Objetivos:	<i>Persistência dos dados dos funcionários</i>		
Dados de entrada:	<i>Nome do funcionário, data de nascimento, endereço, telefone, e-mail, RG, CPF, título de eleitor, estado civil, número da carteira de trabalho</i>		
Operações básicas:	<i>Não tem.</i>		
Dados de saída:	<i>Mensagens de aviso sobre o sucesso (ou não) de CRUD</i>		
Responsável	<i>Alessandra</i>		

4.3 RF003 Manter contrato de trabalho

Descrição: Inserção, busca, alteração e exclusão do contrato de trabalho. O contador deve inserir os dados necessários em um formulário do próprio sistema. Baseando-se

nos dados deste formulário, o sistema deve elaborar contrato de trabalho, seja por tempo determinado ou indeterminado.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF001; RF002; RF005; RF007; RF008; RF009; RF010; RF011; RF012; RF013; RF014</i>			
Objetivos:	<i>Persistência dos dados dos contratos</i>			
Dados de entrada:	<i>Carga horária, horas extras, comissão, duração, experiência, férias, abono pecuniário, vale transporte, vale refeição</i>			
Operações básicas:	<i>Conversões de strings para Float, Int e vice e versa</i>			
Dados de saída:	<i>Mensagens de aviso sobre o sucesso (ou não) de CRUD</i>			
Responsável	<i>Alessandra</i>			

4.4 RF004 Gerar folha de pagamento

Descrição: O contador deve inserir os dados necessários para o preenchimento adequado de um formulário do próprio sistema. Baseando-se neste formulário, o sistema deve gerar e, em seguida, lançar a folha de pagamento baseada no cálculo feito com o uso dos dados (como hora extra e comissões) de cada empregado.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF001; RF002; RF007; RF008; RF009; RF012; RF013</i>			
Objetivos:	<i>Gerar a folha de pagamento de uma empresa</i>			
Dados de entrada:	<i>Valor, mês de referência, data de vencimento</i>			
Operações básicas:	<i>Cálculo de salários, horas extras e comissões de funcionários</i>			
Dados de saída:	<i>Documento com a folha de pagamento</i>			
Responsável	<i>Alessandra</i>			

4.5 RF005 Manter convenção coletiva dos sindicatos

Descrição: O sistema deve possibilitar ao contador armazenar (inserir) a convenção coletiva de sindicato das empresas e, ainda, deve permitir que ele possa modificá-las (alterar) quando ocorrem mudanças. Além disso, o sistema deve criar uma versão digital da convenção coletiva de trabalho de cada sindicato.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF003; RF006; RF012; RF014</i>			

4.6 RF006 Manter sindicatos

Descrição: Inserção, busca, alteração e exclusão de sindicatos.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF001; RF005</i>			
Objetivos:	<i>Persistência dos dados dos sindicatos</i>			
Dados de entrada:	<i>Ramo de atividade, nome, endereço, telefone</i>			
Operações básicas:	<i>Não tem</i>			
Dados de saída:	<i>Mensagens de aviso sobre o sucesso (ou não) de CRUD</i>			
Responsável	<i>Christian</i>			

4.7 RF007 Alertar sobre vencimentos

Descrição: O sistema deverá ter alertas sobre os vencimentos de: férias, contrato de experiência, exame médico, carteira de habilitação, retorno do afastamento e término de aviso prévio.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF003; RF004; RF012; RF014</i>			
Objetivos:	<i>Emitir telas de alertas sobre vencimentos</i>			
Dados de entrada:	<i>Data do sistema</i>			
Operações básicas:	<i>Comparação de datas</i>			
Dados de saída:	<i>Mensagens sobre um aviso, se houver</i>			
Responsável	<i>Kevin</i>			

4.8 RF008 Manter ocorrências

Descrição: Inserção, busca, alteração e exclusão de ocorrências de um funcionário. Uma ocorrência é a anotação de alguma atividade que difere do trabalho em si dentro do contrato de trabalho. Por exemplo, faltas, horas extras e acidentes de trabalho.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			

Requisitos associados:	<i>RF002; RF003; RF004; RF005; RF009; RF010; RF012; RF014</i>
Objetivos:	<i>Persistência dos dados das ocorrências</i>
Dados de entrada:	<i>Data, tipo, justificado, valor</i>
Operações básicas:	<i>Conversão de String para Float</i>
Dados de saída:	<i>Mensagens de aviso sobre o sucesso (ou não) de CRUD</i>
Responsável	<i>Kevin</i>

4.9 RF009 Calcular salário

Descrição: O contador deverá consultar e analisar o plano de cargos e salários e o complemento salarial (ocorrências), além de realizar o cálculo do 13º salário. Se necessário, ele pode preencher um formulário no sistema com os dados salariais, e um arquivo pode ser gerado pelo sistema a partir dos dados preenchidos no formulário.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF002; RF003, RF004, RF008; RF011; RF012</i>			
Objetivos:	<i>Realizar o cálculo de salários dos funcionários</i>			
Dados de entrada:	<i>Funcionário, mês de referência</i>			
Operações básicas:	<i>Cálculo do número de férias, cálculo de abono pecuniário, cálculo do saldo das ocorrências, cálculo do salário</i>			
Dados de saída:	<i>Salário do funcionário calculado</i>			
Responsável	<i>Kevin</i>			

4.10 RF010 Realizar tarefas de rescisão

Descrição: O contador deverá, sob demanda, realizar avisos prévios, cálculo normal e complementar, GRFC e/ou requerimento de seguro desemprego, em caso de rescisão de contrato de algum empregado de alguma empresa.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF001; RF002; RF003; RF008; RF014</i>			
Objetivos:	<i>Rescindir contrato de um funcionário com uma empresa</i>			
Dados de entrada:	<i>Data da rescisão, justificado, valor a ser pago (se justificado=true)</i>			
Operações básicas:	<i>Cálculo do pagamento de rescisão</i>			
Dados de saída:	<i>Pagamento da rescisão</i>			

Responsável	Kevin e Alessandra
--------------------	--------------------

4.11 RF011 Realizar tarefas de admissão

Descrição: Baseando-se em dados e consultas preenchidos e registrados em formulário pelo contador, o sistema deve gerar contrato de experiência, solicitação de vale-transporte, acordo de compensação ou prorrogação e declaração de dependentes e outros tipos de relatórios.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF001; RF002; RF003; RF009</i>			
Objetivos:	<i>Vincular um funcionário a uma empresa</i>			
Dados de entrada:	<i>Dados de: funcionário, empresa e contrato</i>			
Operações básicas:	<i>Conversões de tipos da linguagem</i>			
Dados de saída:	<i>Contrato, solicitação de vale-transporte e vale-refeição, acordo de compensação ou prorrogação e declaração de dependentes e outros tipos de relatórios</i>			
Responsável	<i>Alessandra</i>			

4.12 RF012 Calcular férias

Descrição: O contador deve calcular as férias normais e coletivas, vigentes no contrato de trabalho. Os resultados devem ser registrados no sistema.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF002; RF003; RF004; RF005; RF007; RF008; RF009</i>			
Objetivos:	<i>Calcular o número de dias que um funcionário tem de férias</i>			
Dados de entrada:	<i>Números de ocorrências (faltas) não justificadas e número de férias estabelecidas no contrato</i>			
Operações básicas:	<i>Cálculo do número de férias</i>			
Dados de saída:	<i>Número de férias</i>			
Responsável	<i>Kevin</i>			

4.13 RF013 Calcular abono pecuniário

Descrição: O contador deve realizar o gerenciamento de vale-transporte e vale-refeição e, enquanto editar os dados do empregado que estão no sistema, deve inserir ou alterar os dados de gerenciamento de cada vale.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF002; RF003; RF004</i>			
Objetivos:	<i>Calcular abono pecuniário de funcionário</i>			
Dados de entrada:	<i>Número de férias de um funcionário</i>			
Operações básicas:	<i>Cálculo do abono pecuniário</i>			
Dados de saída:	<i>Valor do abono pecuniário</i>			
Responsável	<i>Christian</i>			

4.14 RF014 Manter aviso prévio

Descrição: Inclusão, busca, alteração e exclusão de aviso prévio. Só é incluso quando ocorre uma rescisão de contrato, quando é especificado seu tipo e seu tempo de duração. E só é excluído quando o tempo de duração é expirado.

Prioridade:	Essencial	X	Importante	Desejável
Ator(es):	<i>Contador.</i>			
Requisitos associados:	<i>RF002; RF003; RF005; RF007; RF008; RF010</i>			
Objetivos:	<i>Persistência de dados de aviso prévio</i>			
Dados de entrada:	<i>Data de aviso, data de rescisão, justificado, motivo</i>			
Operações básicas:	<i>Não tem</i>			
Dados de saída:	<i>Mensagens de aviso sobre o sucesso (ou não) de CRUD</i>			
Responsável	<i>Kevin</i>			

5 Requisitos Não-Funcionais

5.1 Requisitos de Desempenho

- NF001 O sistema deve possuir o MANAD;
- NF002 O tempo de processamento de uma operação de Cadastro de Cliente (RF001) e de Registro de Empregados (RF002) não deve exceder três segundos;
- NF003 O tempo de processamento de uma operação de qualquer um dos outros requisitos funcionais (RF003 a RF006) não deve exceder cinco segundos;

5.2 Requisitos de Segurança

- NF004 O sistema, ao ser inicializado, deve exigir um nome de usuário e senha, para que apenas o contador tenha acesso às funções do sistema;
- NF005 O sistema deve ter capacidade de salvar automaticamente as alterações realizadas no sistema a cada dez minutos, caso o contador opte por essa função;
- NF006 O sistema, após sofrer uma queda indesejada (queda de energia, fechamento do sistema por variados motivos), ao ser reinicializado, deve perguntar ao usuário se deseja continuar a última operação recorrente;

5.3 Atributos de Qualidade do Software

- NF007 O sistema deve rodar em microcomputadores da linha IBM PC que possuam no mínimo um processador de 1 GHZ, memória RAM mínima de 512 Megabits e que execute em plataforma *Linux*;

6 Modelo de Casos de Uso

O Diagrama de Casos de Uso tem o objetivo de auxiliar a comunicação entre os analistas e o cliente, descrevendo um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário.

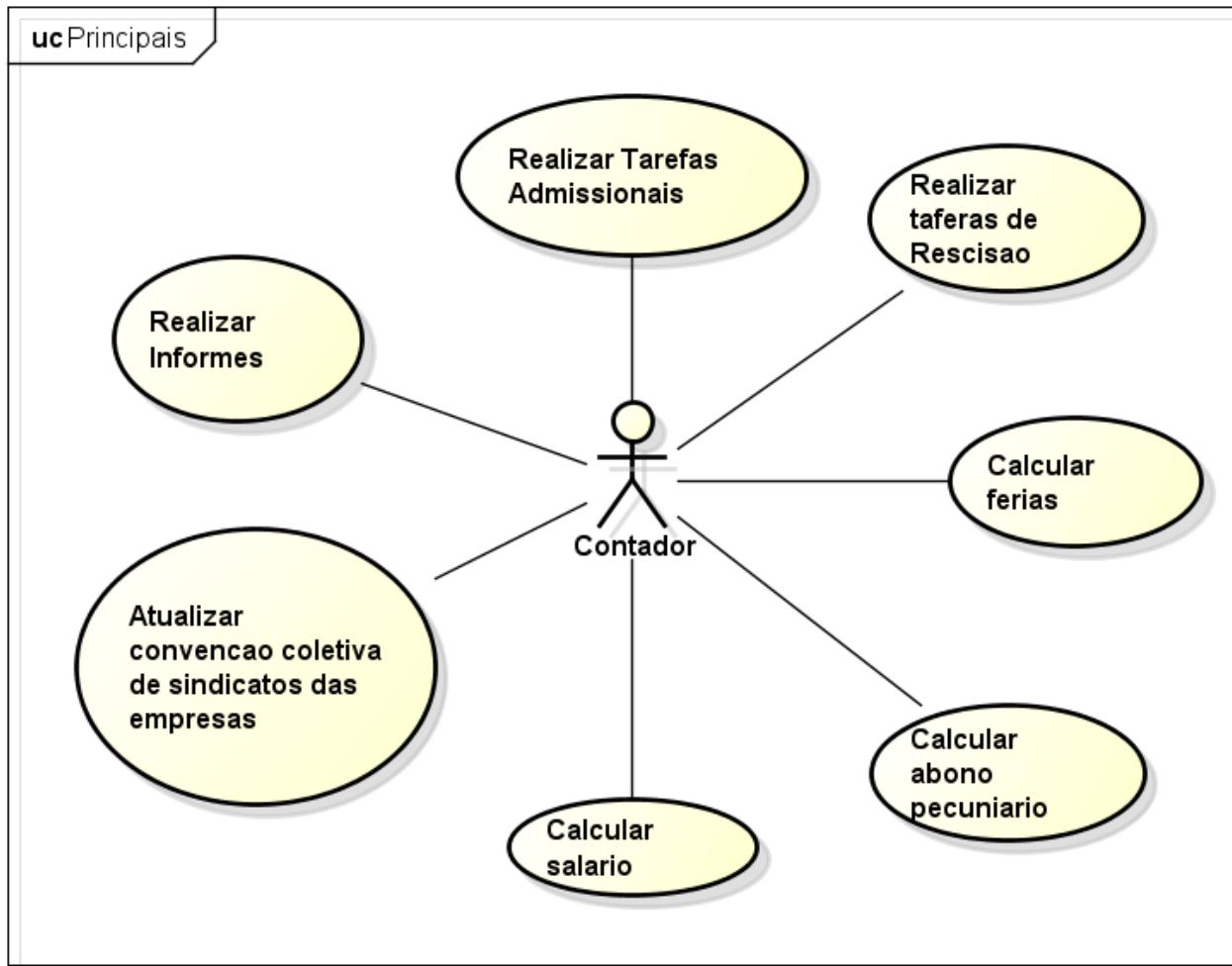


Figura 1 - Modelo de Casos de Uso do Sistema Contábil Ribanceira Recursos Humanos

Na Figura 1 temos os principais processos do sistema: Realizar Tarefas Admissionais, Atualizar Convenção Coletiva de Sindicatos das Empresas, Realizar Tarefas de Rescisão, Calcular férias e abono pecuniário, Calcular salário e Realizar Informes. Todos esses processos são realizados pelo único ator, o Contador. Os casos de uso principais são descritos a seguir.

6.1 Descrição dos Casos de Uso Principais

6.1.1 Caso de Uso: Realizar tarefas de admissão.

Identificador: RF011

Descrição: Apoia nas tarefas de (re)integração de um funcionário a uma empresa.

Ator: Contador.

Pré-Condição: A empresa contratante e o funcionário a ser contratado devem estar cadastrados no sistema.

Pós-Condição: Documentações de admissão prontas.

Curso Normal:

- 1) O Contador seleciona a empresa contratante e o funcionário a ser contratado;
- 2) O Contador informa se o contratado é novo na empresa;
- 3) O Contador informa se o contratado deseja solicitar Vale-Transporte (VT) e/ou Vale-Refeição (VR);
- 4) O Contador informa se o contratado possui dependentes;
- 5) O Contador informa se o contrato é de experiência (tempo determinado);
- 6) O Contador calcula e cadastra o salário do empregado novo;
- 7) O sistema gera as solicitações (se houverem) e contrato de trabalho.

Tratamento das Exceções:

- 1) Se a empresa e/ou o funcionário não estiverem cadastrados no sistema, o sistema pergunta se o Contador deseja realizar o cadastro ou se deseja encerrar a operação;
- 2) Se o contratado não é empregado novo, o Contador informa se o contratado quer acordo de compensação ou de prorrogação;
- 3) Se o Contador não preencher todos os campos necessários, o sistema alerta-o sobre a falta de dados e verifica se ele quer completar os campos vazios ou cancelar a operação.

6.1.2 Caso de Uso: Manter convenção coletiva de sindicatos das empresas.

Identificador: RF005

Descrição: Dá suporte às eventuais alterações da convenção coletiva de sindicatos das empresas.

Ator: Contador

Pré-Condição: Convenção coletiva já ter sido inserido no sistema.

Pós-Condição: Convenção coletiva atualizada.

Curso Normal:

- 1) O Contador seleciona a convenção coletiva que será alterada;
- 2) O Contador gera a versão digital para impressão convenção coletiva que será alterada;
- 3) O Contador realiza as alterações manualmente da convenção coletiva;

- 4) O Contador salva as alterações feitas.

Tratamento de Exceções:

- 1) Se a convenção coletiva a ser alterada ainda não está presente no sistema, este pergunta se o Contador deseja importar a convenção coletiva ou cancelar a operação;
- 2) Se ocorrer uma falha na geração da versão digital, o sistema alerta sobre o erro e verifica se o Contador deseja tentar novamente;
- 3) Se ocorrer uma falha durante a alteração, o sistema alerta sobre do que se trata o erro e pergunta se ele deseja salvar o progresso até aquele momento ou se deseja cancelar a operação;
- 4) Se ocorrer falha no momento de salvar, o sistema avisa sobre a falha e pergunta se o Contador deseja tentar novamente ou se deseja cancelar a operação.

6.1.3 Caso de uso: Realizar tarefas de rescisão**Identificador: RF010**

Descrição: gerenciar o empregado que será rescindido da empresa;

Ator(es): contador;

Pré-condição: o contador deve ter os registros do funcionário que será rescindido e da empresa contratante no sistema contábil;

Pós-condição: geração dos relatórios conforme os dados procedidos.

Curso Normal:

- 1) O Contador verifica se a empresa e o empregado estão registrados no sistema;
- 2) O Contador consulta os dados de rescisão do empregado que será rescindido;
- 3) O Contador realiza o aviso prévio ao empregado que será demitido;
- 4) O Contador reaiza os cálculos normais e complementares conforme o tempo de trabalho do empregado e seus benefícios.

Tratamento das Exceções:

- 1) O Contador não encontra os dados da empresa ou do empregado:
 - a. O sistema emite uma mensagem de erro e pede para verificar os dados inseridos.
- 2) O empregado consultado será demitido sem justa causa:
 - a. O Contador gera o GRFC que será destinado ao recolhimento para o FGTS e da Contribuição Socail;
 - b. O Contador calcula a multa que a empresa deverá pagar ao empregado.
- 3) O empregado consultado possui vínculo de pelo menos 18 meses:
 - a. O Contador gera o requerimento de seguro desemprego.
- 4) O empregado consultado pediu a demissão e não conseguiu aviso prévio:
 - a. O Contador calcula a multa que será descontada do pagamento de rescisão.

6.1.4 Casos de uso: Calcular férias e Calcular abono pecuniário

Identificador: RF012 e RF013

Descrição: o contador calculará as férias normais e coletivas, além do complemento de férias e abono pecuniário;

Autor(es): contador;

Pré-condição: o contador deve ter os registros do funcionário que entrará em férias;

Pós-condição: os resultados devem ser registrados no sistema.

Curso Normal:

- 1) Contador verifica se a empresa e o empregado estão registrados no sistema;
- 2) Contador verifica os dias de férias previsto pela legislação;
- 3) Contador verifica o início das férias determinado pela empresa;
- 4) Contador calcula as datas e prazos de férias do empregado;
- 5) Contador atualiza os dias de trabalho do empregado.

Tratamento das Exceções:

- 1) Contador não encontrou os dados da empresa ou do empregado
 - a. Sistema emite mensagem de erro e pede para verificar os dados inseridos.
- 2) O empregado solicitou o abono pecuniário
 - a. Contador remunera as datas de férias decididas previamente
 - b. Contador calcula a remuneração de férias com acréscimo de 1/3 de férias da legislação trabalhista.
- 3) O empregado faltou alguns dias e precisa repor estas faltas
 - a. Contador reduz as férias do empregado, conforme o período aquisitivo.

6.1.5 Caso de uso: Calcular salário

Identificador: RF009

Descrição: Calcula o salário de algum funcionário

Autor(es): Contador

Pré-condição: Funcionário cadastrado no sistema

Pós-condição: Salário calculado

Curso normal:

- 1) Contador procura o funcionário no sistema;
- 2) Contador consulta a base salarial da função do funcionário nessa empresa;
- 3) A folha de pagamento é lançada pelo sistema.

Tratamento de exceções:

- 1) Funcionário não cadastrado/não encontrado:
 - a. Emite alerta de erro informando falha na busca.
- 2) Base salarial não encontrada:
 - a. Emite alerta de erro informando falha na busca.
- 3) Falta de informações para a emissão da folha de pagamento:
 - a. Emite alerta de erro informando quais são as informações faltantes;
 - b. Requer ao contador que insira essas informações.

6.1.6 Caso de uso: Realizar informes

Identificador: RFXXX

Descrição: Realiza os informes de rendimento, RAIS, DIRF, SEFIP e CAGED

Ator(es): Contador

Pré-condição: Informe solicitado

Pós-condição: Informe impresso

Curso normal:

- 1) Obtém as informações necessárias para gerar o informe solicitado;
- 2) Preenche o modelo do informe com as informações obtidas;
- 3) Imprime o informe.

Tratamento de exceções:

- 1) Informações insuficientes:
 - a. Alerta de erro solicitando ao contador as informações faltantes.

6.2 Tabela de Conceitos e Consultas

Os casos de uso de manutenção são organizados como mostrado na Tabela 1. Note que nela aparecem todos os conceitos que aparecem no modelo de objetos de negócio (Figura X, na seção X). Para cada um deles aparecem as operações de manutenção que serão realizadas no sistema e em qual caso de uso será feita determinada operação (coluna Referência Cruzada). O Contador (ator), por exemplo, não pode inserir nem remover a Convenção Coletiva de Trabalho. Isso se deve ao fato de que a Convenção Coletiva é inserida ou removida ao se inserir ou remover um Sindicato. O Contador pode apenas Alterar ou Consultar a Convenção Coletiva o que é feito no Caso de Uso de identificador RF005.

Conceito	Inserir	Alterar	Remover	Consultar	Observação	Referência Cruzada
Cliente	X	X	X	X		RF011, RF014
Empregado	X	X	X	X	Não é possível remover um Empregado enquanto tiver vínculo com um Contrato	RF011, RF010, RF012, RF009, RF014
Contrato	X	X	X	X		RF011, RF005, RF010, RF009, RF014
Salário	X	X	X	X	Não é possível remover um Salário enquanto tiver vínculo com Empregado e um Contrato	RF011, RF012, RF009
Informes	X	X	X	X		RF014
Férias	X	X	X	X	As Férias podem variar conforme a presença do Empregado	RF005, RF012, RF014
Vales	X	X	X	X		RF011, RF009
Convenção Coletiva de Trabalho		X		X	Não é possível inserir ou remover uma Convenção Coletiva de Trabalho, esta é determinada pelos Sindicatos	RF005
SEFIP	X	X	X	X		RF014

Tabela 1 – Tabela de Conceitos e Operações do Sistema Contábil Ribanceira Recursos Humanos

Uma consulta é uma operação de acesso a um conjunto de informações armazenadas no sistema, por exemplo, em sua base de dados. Os resultados de uma consulta são estruturados em relatórios, por exemplo, um Relatório dos pagamentos realizados aos empregados por parte da empresa. As operações de consulta não alteram o estado do sistema. Estas estão representadas na Tabela 2. O número de relatórios que podem ser retirados do sistema é grande, portanto não é interessante representar essas operações no modelo de casos de uso principais. Na Tabela 2, por exemplo, a consulta do porcentual do salário do empregado que o empregador é obrigado a depositar na conta do empregado na Caixa Econômica Federal (para a realização de informes caso de uso de identificador RF014) gera um relatório, o FGTS.

Nome	Referência
Folha de Pagamento	RF012, RF009, RF014
Base Salarial	RF011, RF005, RF009
Férias Normais	RF005, RF012
Férias Coletivas	RF005, RF012
FGTS	RF014
INSS	RF014
DIRF	RF014

Tabela 2 – Tabela de Consultas do Sistema Contábil Ribanceira Recursos Humanos

Todos os estes casos de uso podem ser representados em apenas um diagrama, como segue nas próximas quatro figuras:

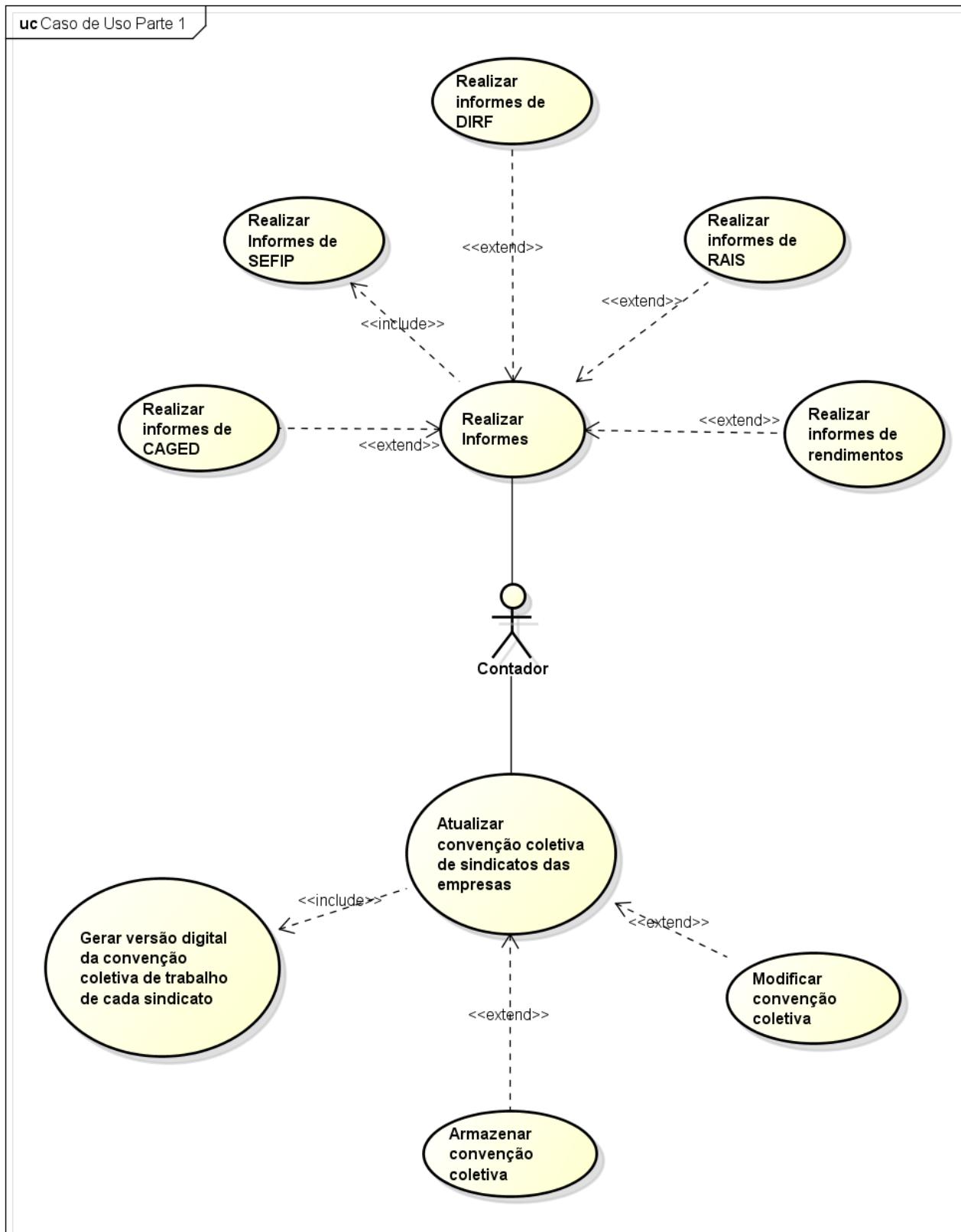


Figura 2 - Diagrama de Casos de Uso do Sistema Contábil Ribanceira Recursos Humanos Parte 1

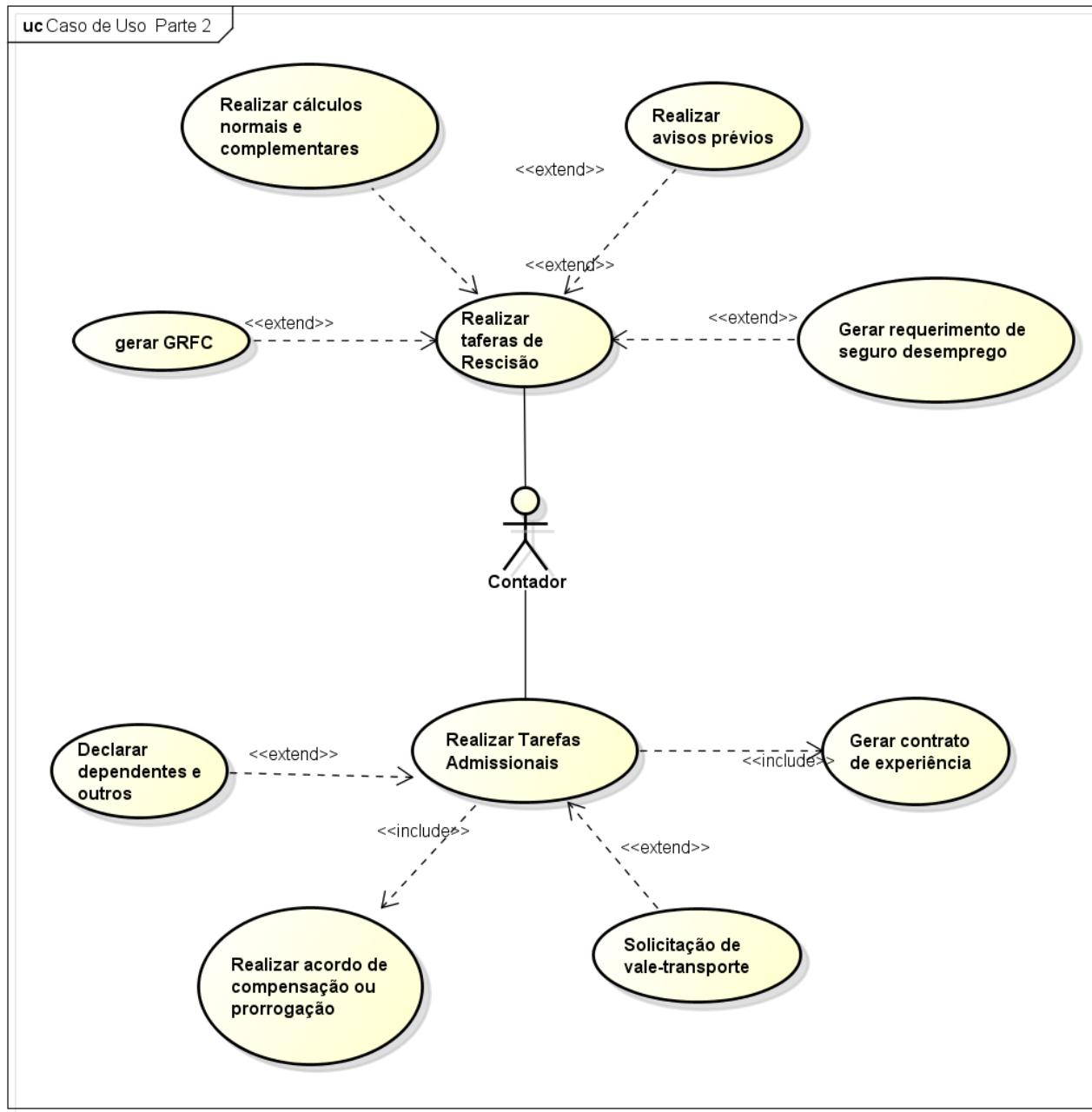


Figura 3 - Diagrama de Casos de Uso do Sistema Contábil Ribanceira Recursos Humanos Parte 2

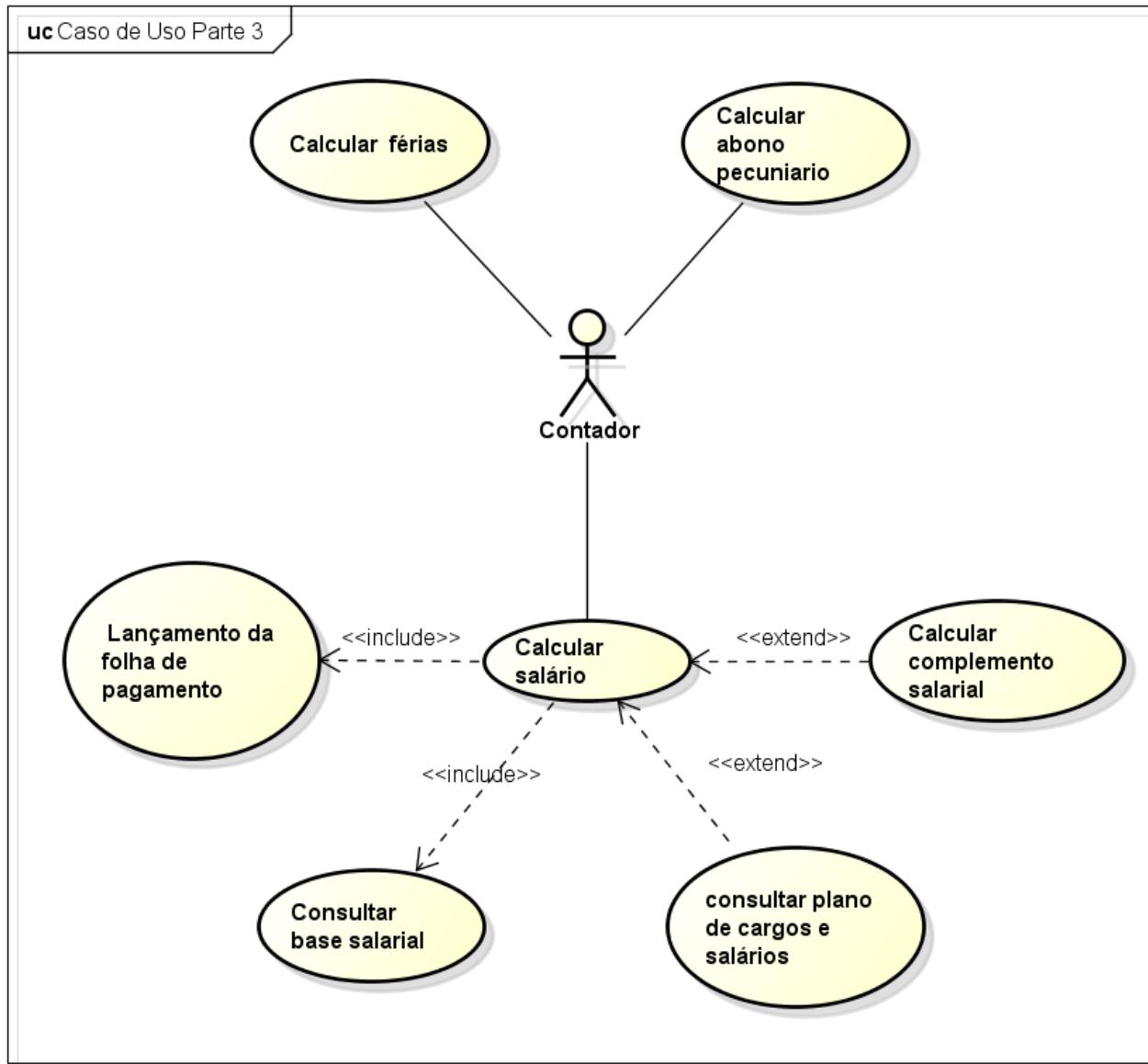


Figura 4 - Diagrama de Casos de Uso do Sistema Contábil Ribanceira Recursos Humanos Parte 3

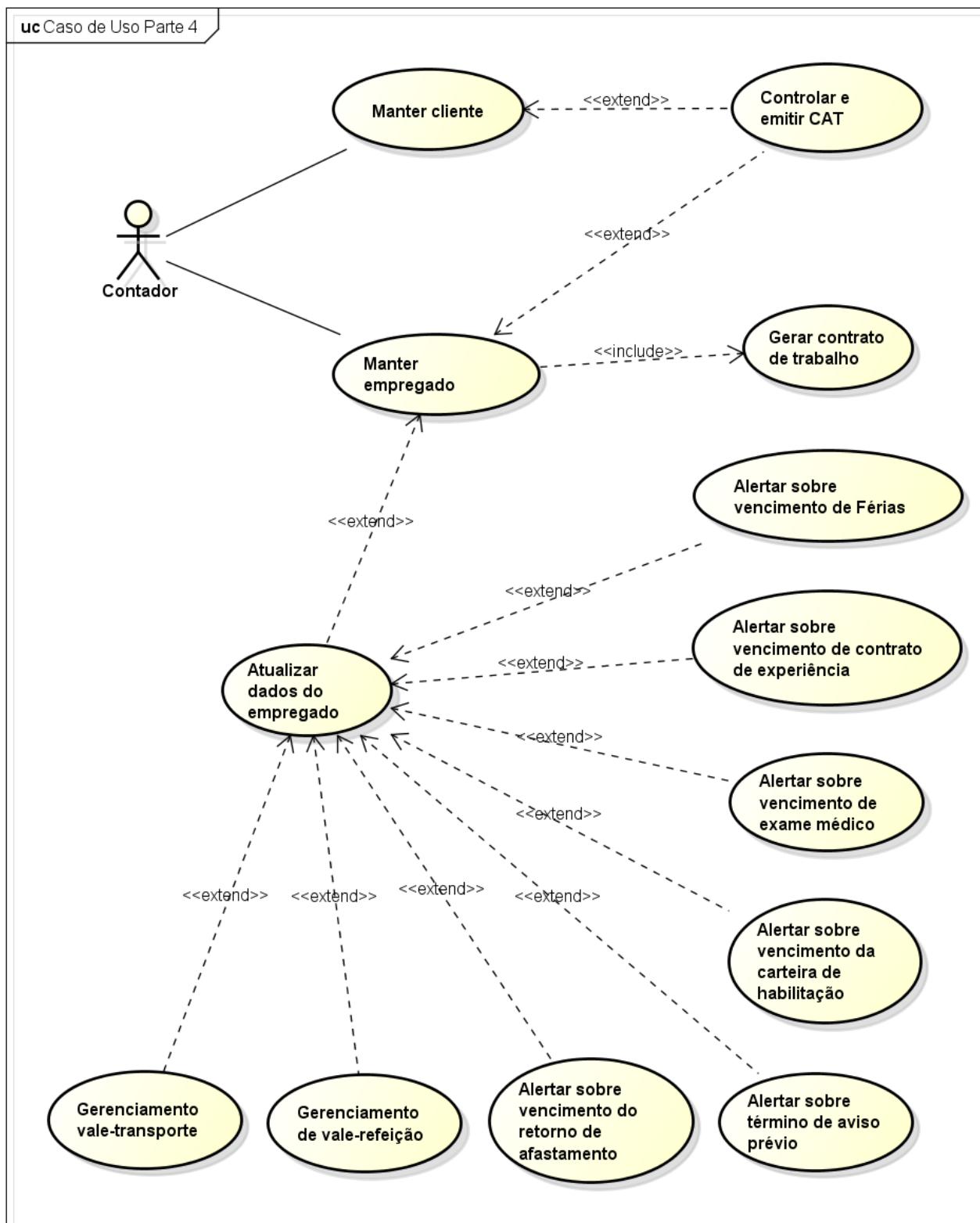


Figura 5 - Diagrama de Casos de Uso do Sistema Contábil Ribanceira Recursos Humanos Parte 4

7 Diagrama de Pacotes

O diagrama de pacotes a ser apresentado procura explicitar a análise arquitetural do sistema. Dividido entre camada de aplicação específica e camada de aplicação geral.

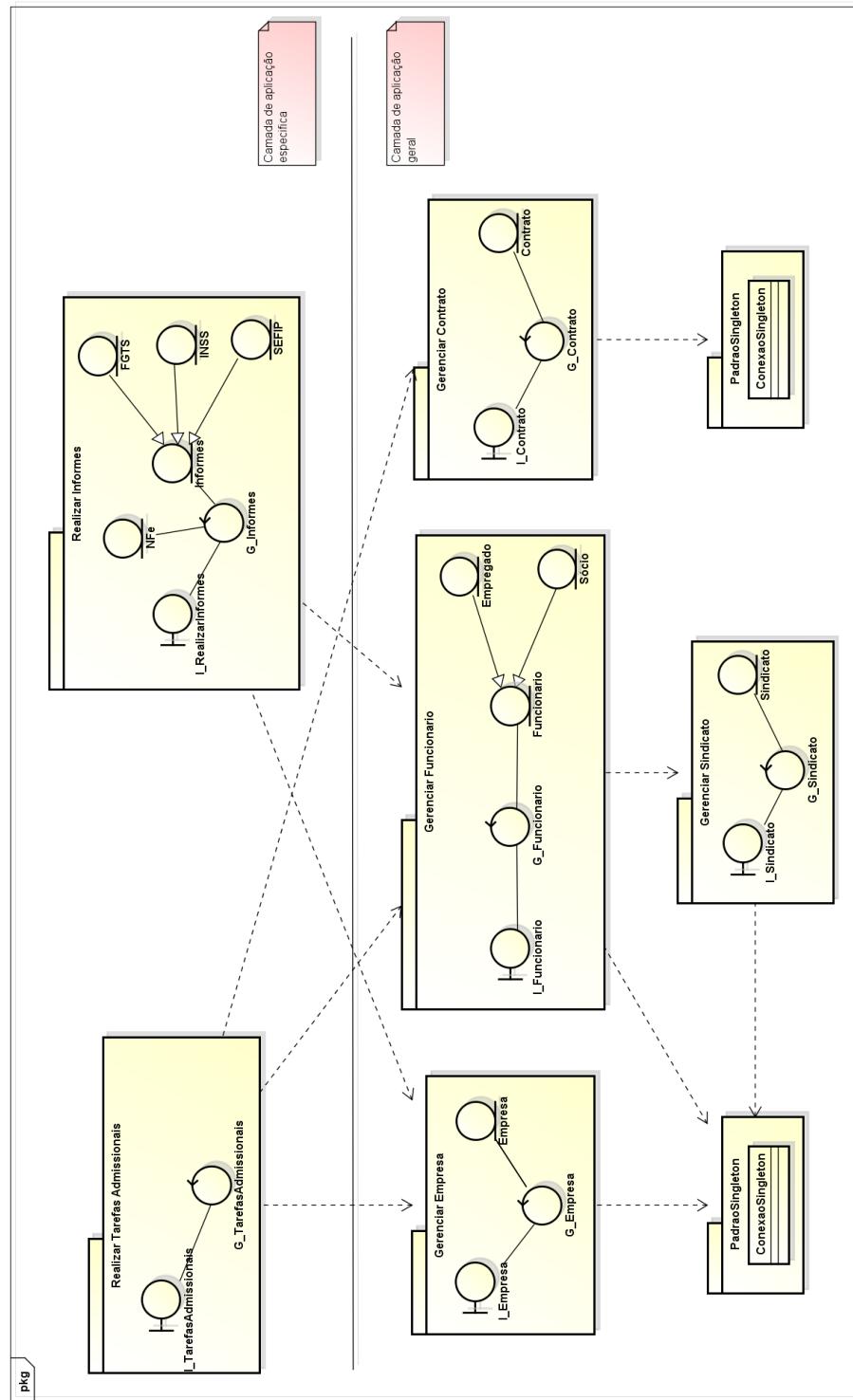


Figura 6 - Diagrama de Pacotes do Sistema Contábil Ribanceira Recursos Humanos Parte 1

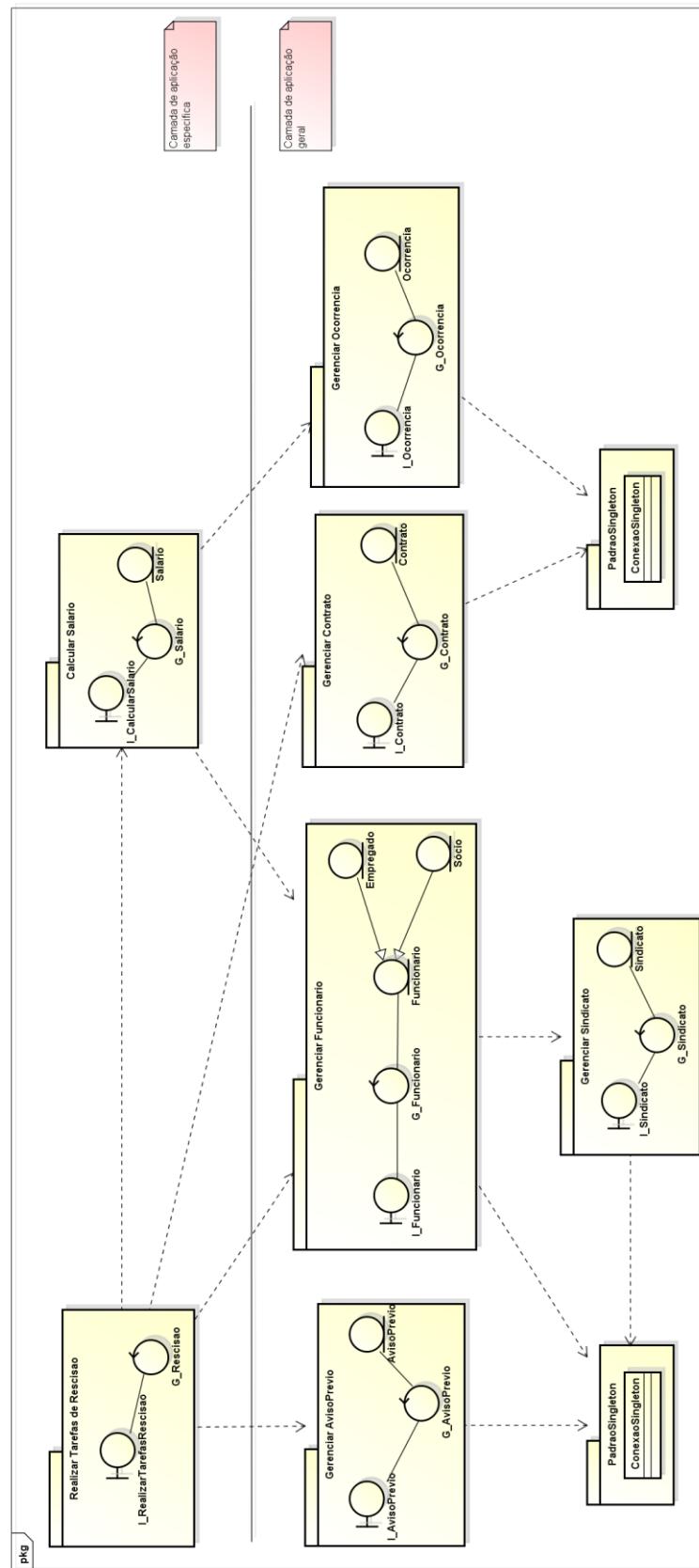


Figura 7 - Diagrama de Pacotes do Sistema Contábil Ribanceira Recursos Humanos Parte 2

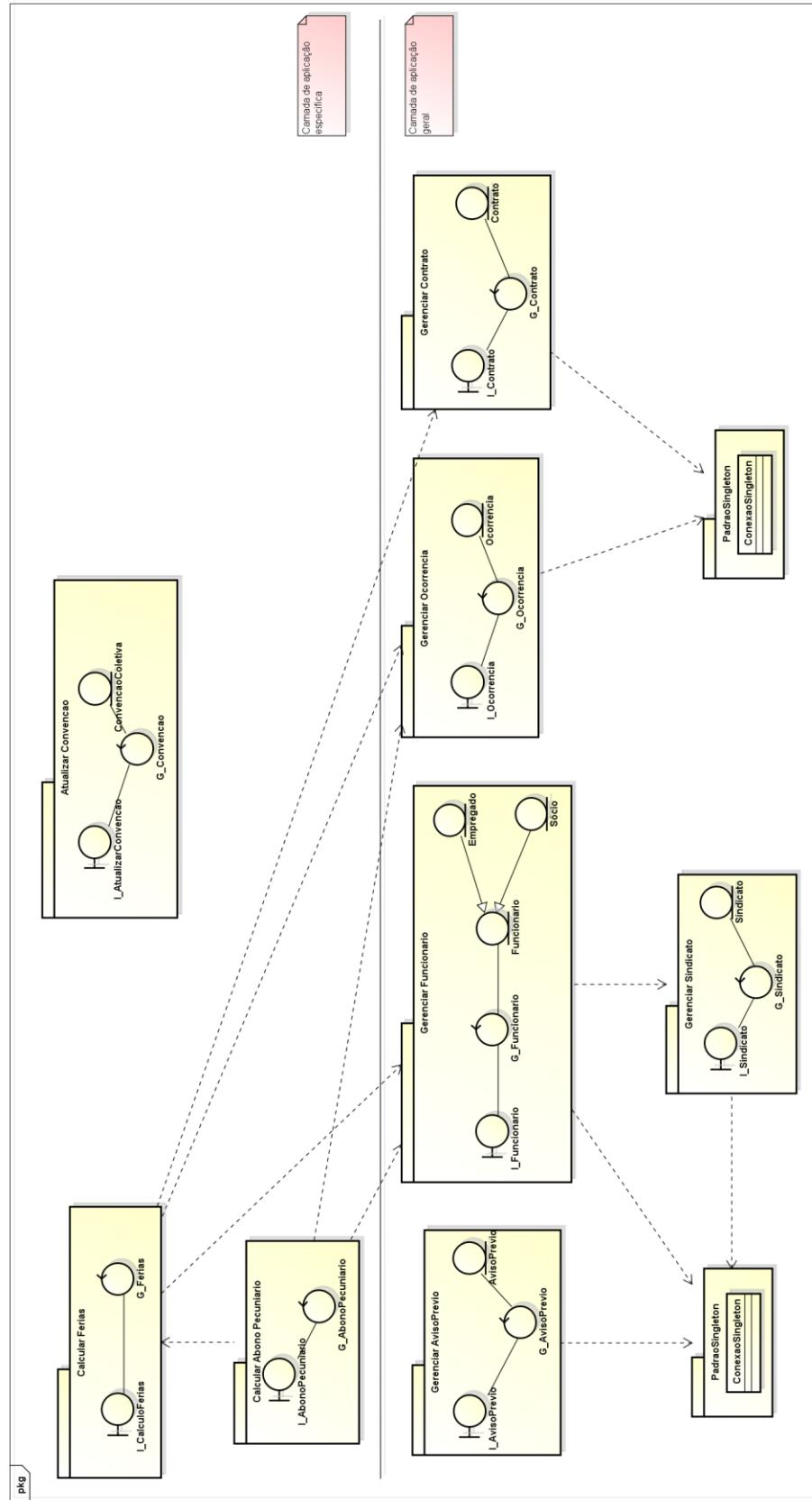


Figura 8 - Diagrama de Pacotes do Sistema Contábil Ribanceira Recursos Humanos Parte 3

8 Diagrama de Classes

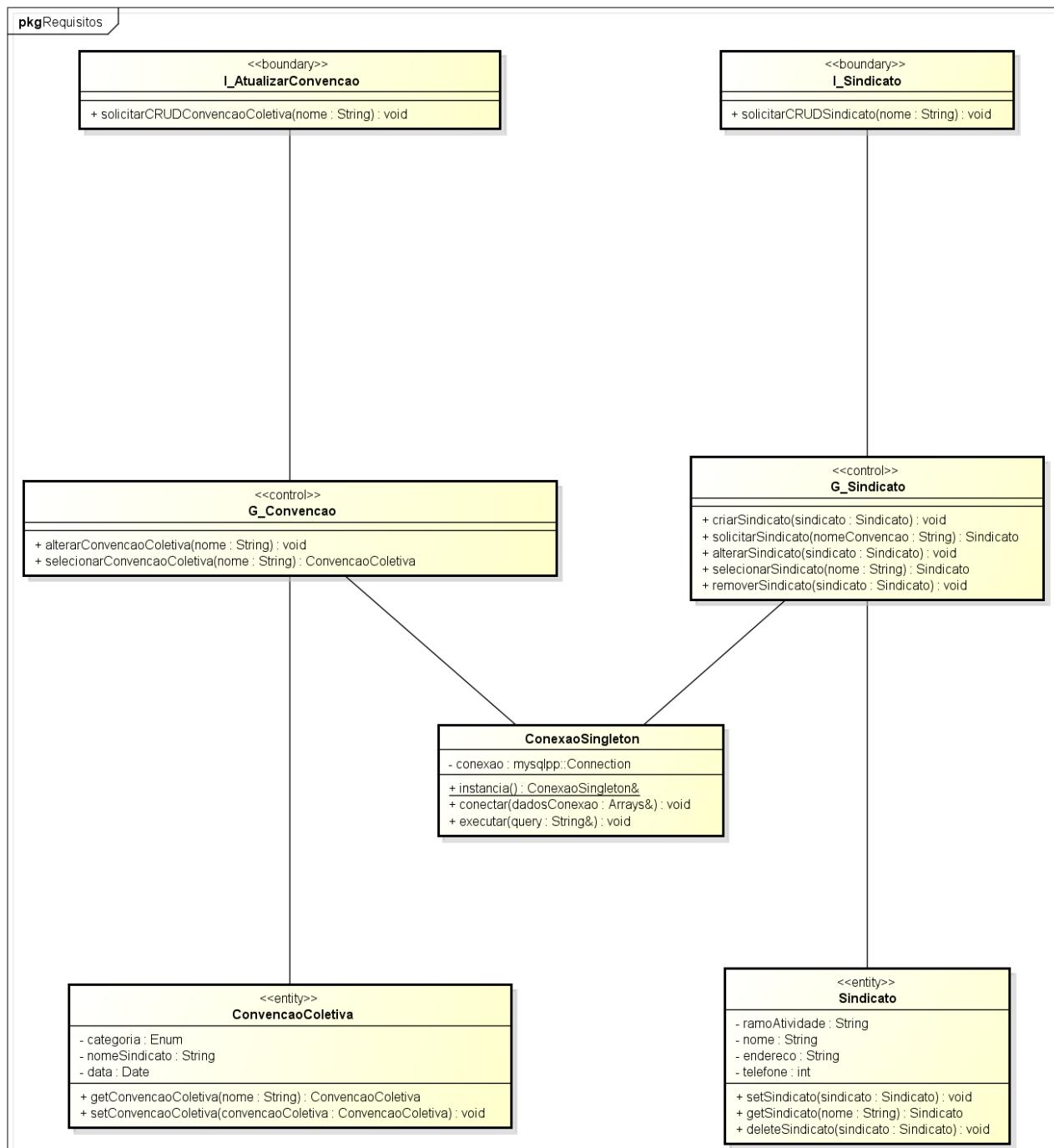


Figura 9 - Diagrama de Classes do Sistema Contábil Ribanceira Recursos Humanos Parte 1

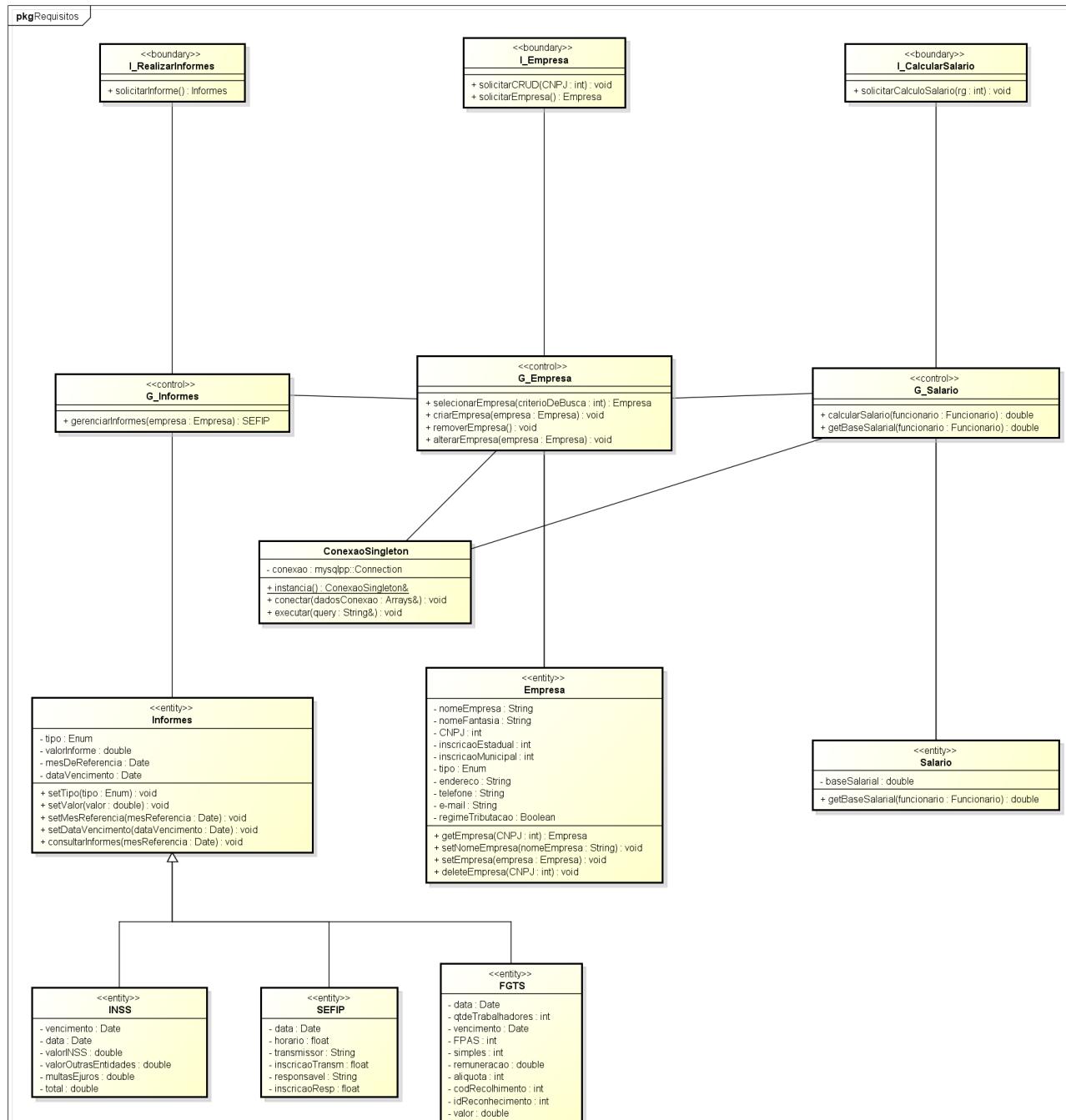


Figura 10 - Diagrama de Classes do Sistema Contábil Ribanceira Recursos Humanos Parte 2

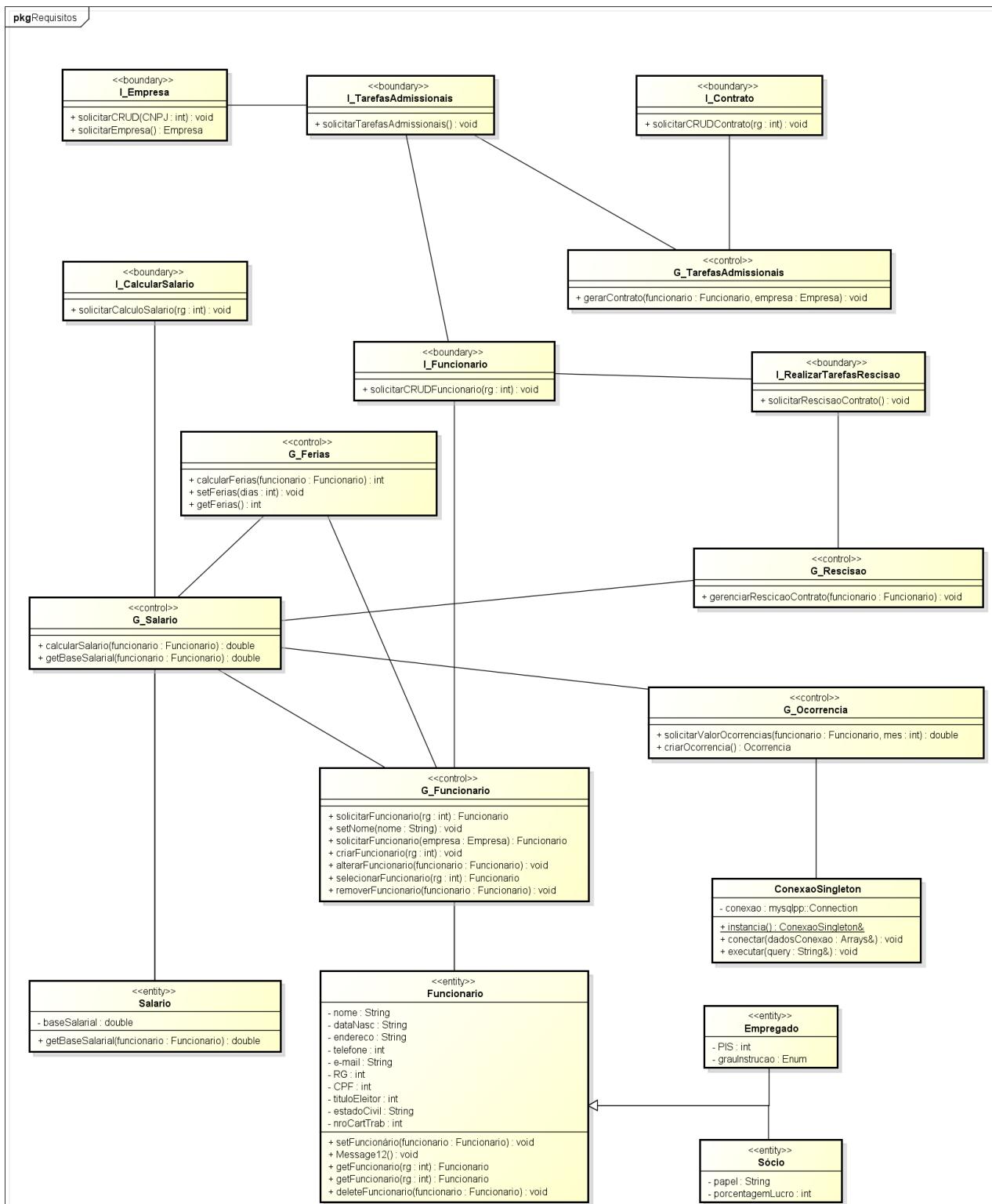


Figura 11 - Diagrama de Classes do Sistema Contábil Ribanceira Recursos Humanos Parte 3

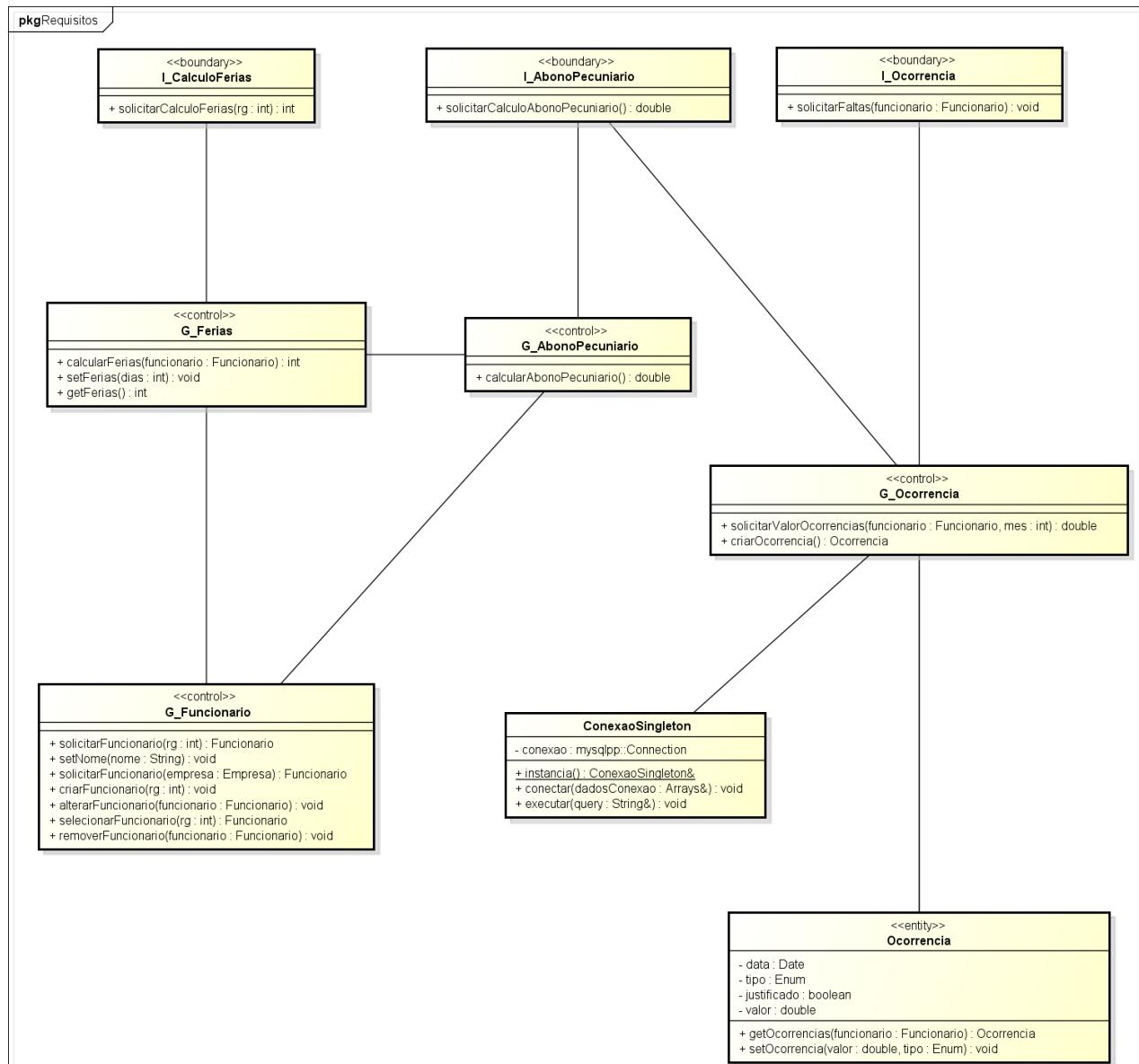


Figura 12 - Diagrama de Classes do Sistema Contábil Ribanceira Recursos Humanos Parte 4

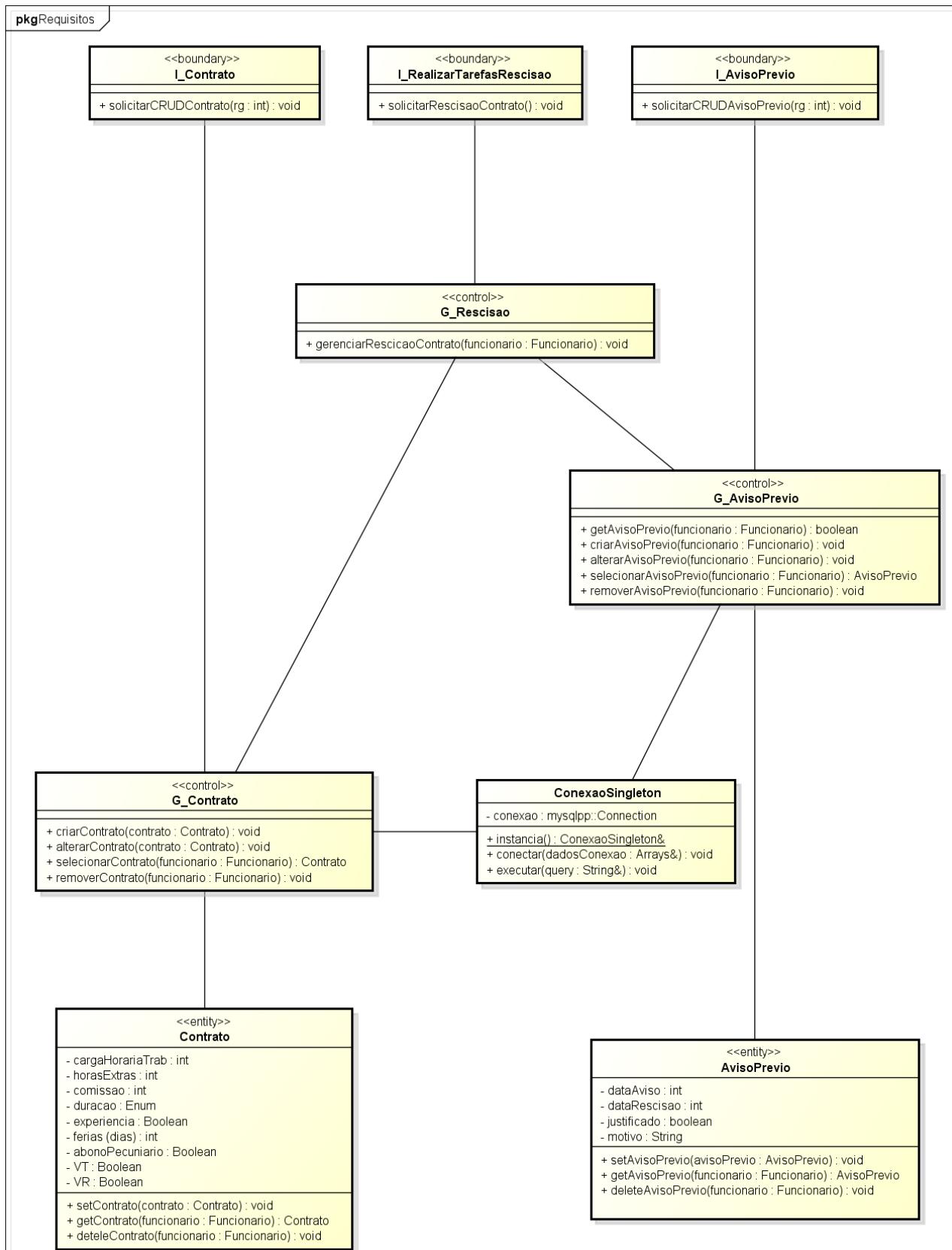


Figura 13 - Diagrama de Classes do Sistema Contábil Ribanceira Recursos Humanos Parte 5

Foram utilizados os frameworks MySQL++ e o Gtkmm (seção 10.3). A classe ConexaoSingleton é uma herança do mysql::conector. O mysql::conector é uma classe do MySQL++ utilizada para realizar a conexão com o banco de dados.

Já o Gtkmm está sendo utilizado para a criação das interfaces (seção 11). O menu principal é uma herança de Gtk::Window e os restantes das telas de Gtk::Dialog. Os frameworks e suas ligações que acabaram de ser descritas podem ser observadas na Figura 14.

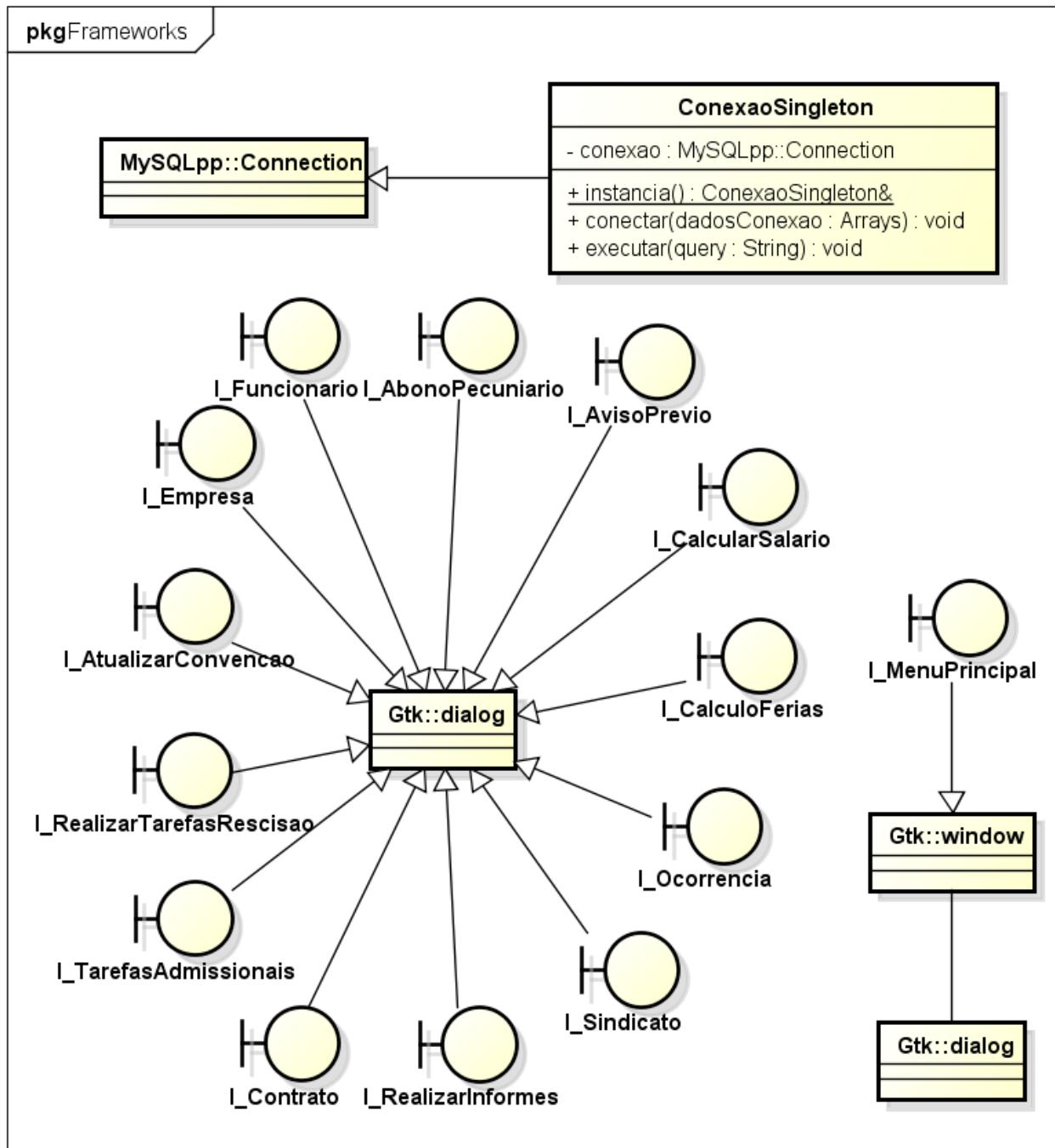


Figura 14 - Diagrama de Classes do Sistema Contábil Ribanceira Recursos Humanos Parte 6

9 Diagrama de Comunicação

Nos diagramas a seguir, as classes estão estereotipadas, assim como o diagrama de pacotes (seção 7). Além disso, ficam evidentes as trocas de mensagens que ocorrem entre as diversas classes para que uma função do sistema seja realizada. Encontram-se no padrão do MVC (*model-view-control*), ou seja, *boundary* (fronteira), *control* (controle) e *entity* (entidade).

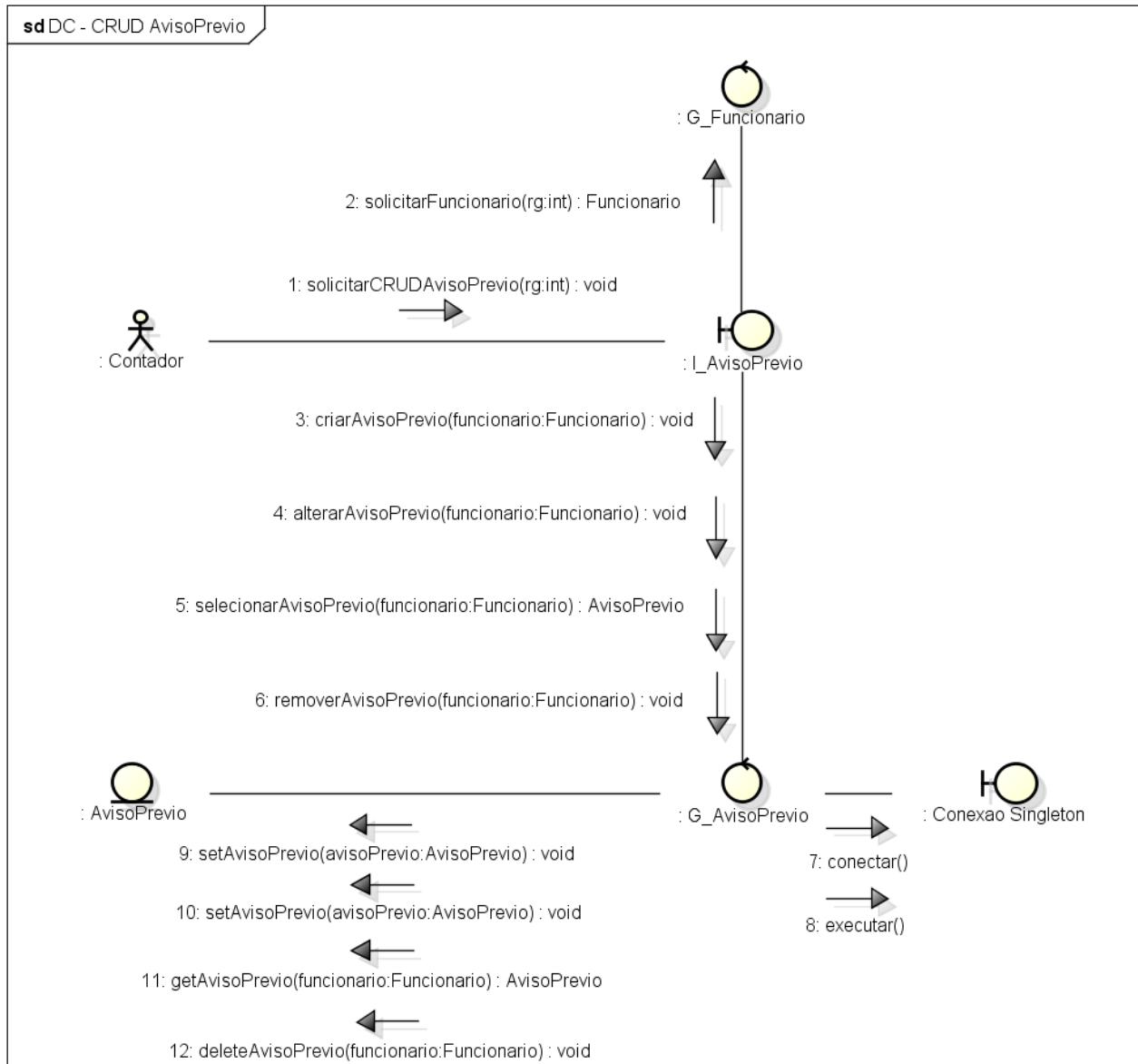


Figura 15 - Diagrama de Comunicação de CRUD Aviso Prédio

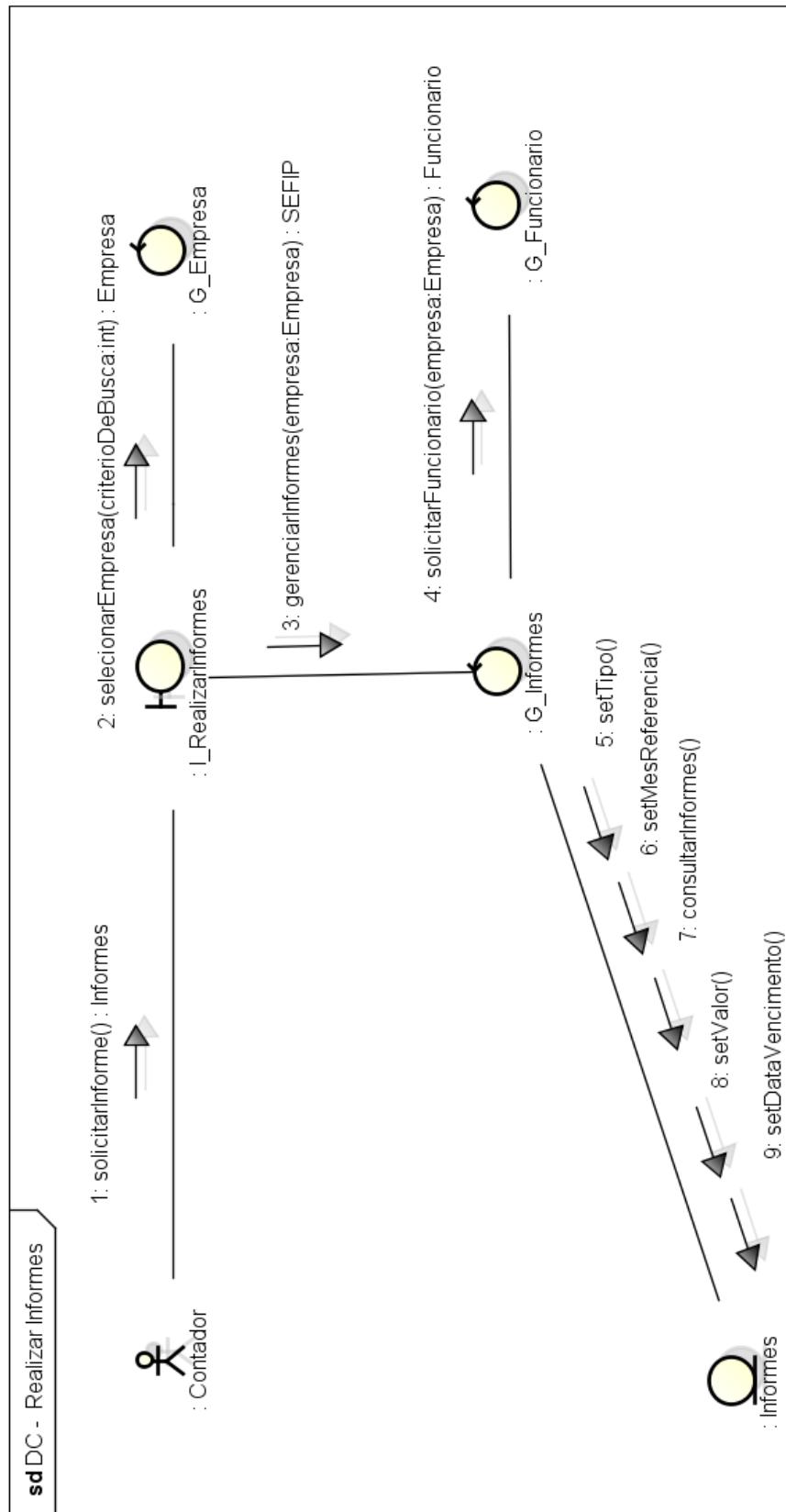


Figura 16 - Diagrama de Comunicação de Realizar Informes

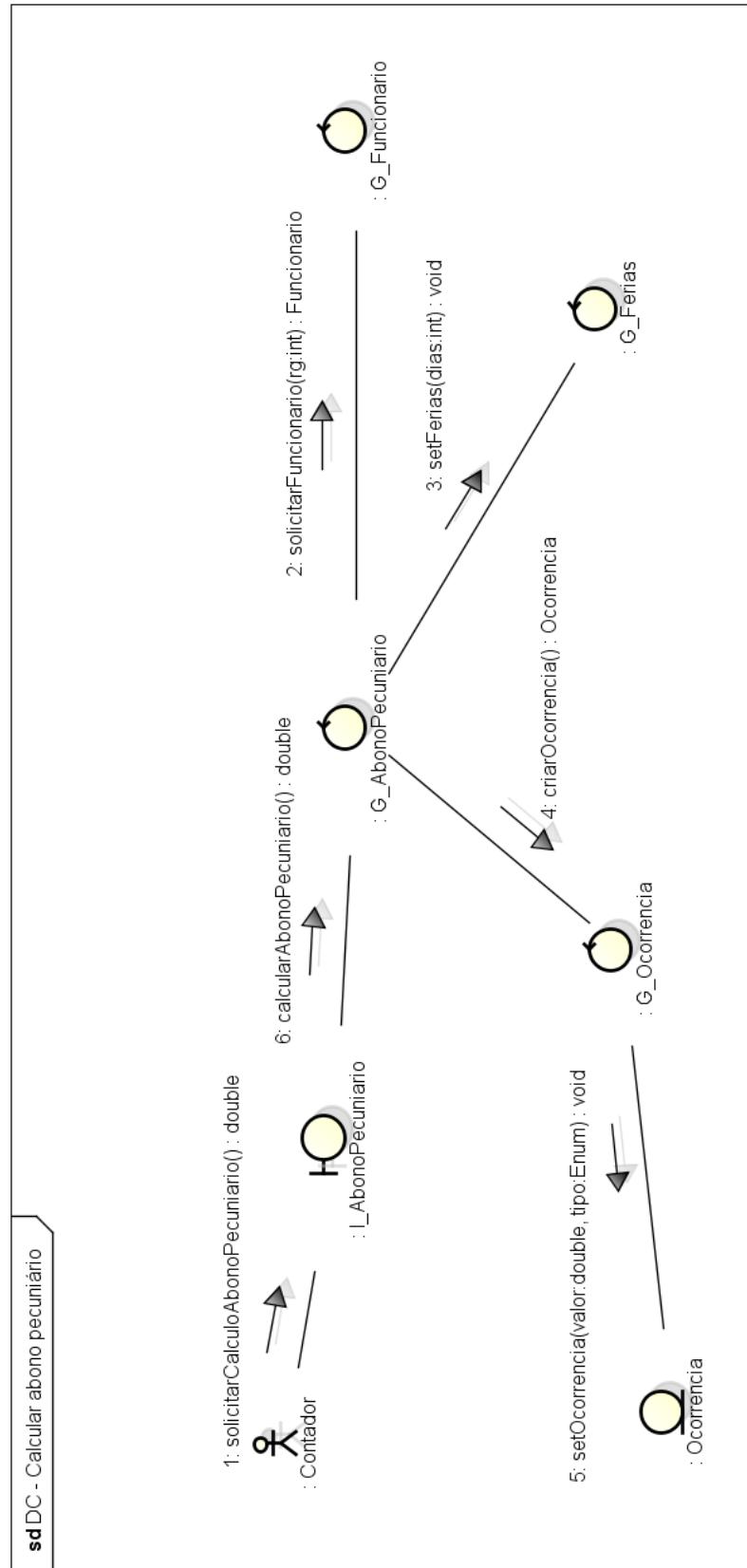


Figura 17 - Diagrama de Comunicação de Calcular abono pecuniário

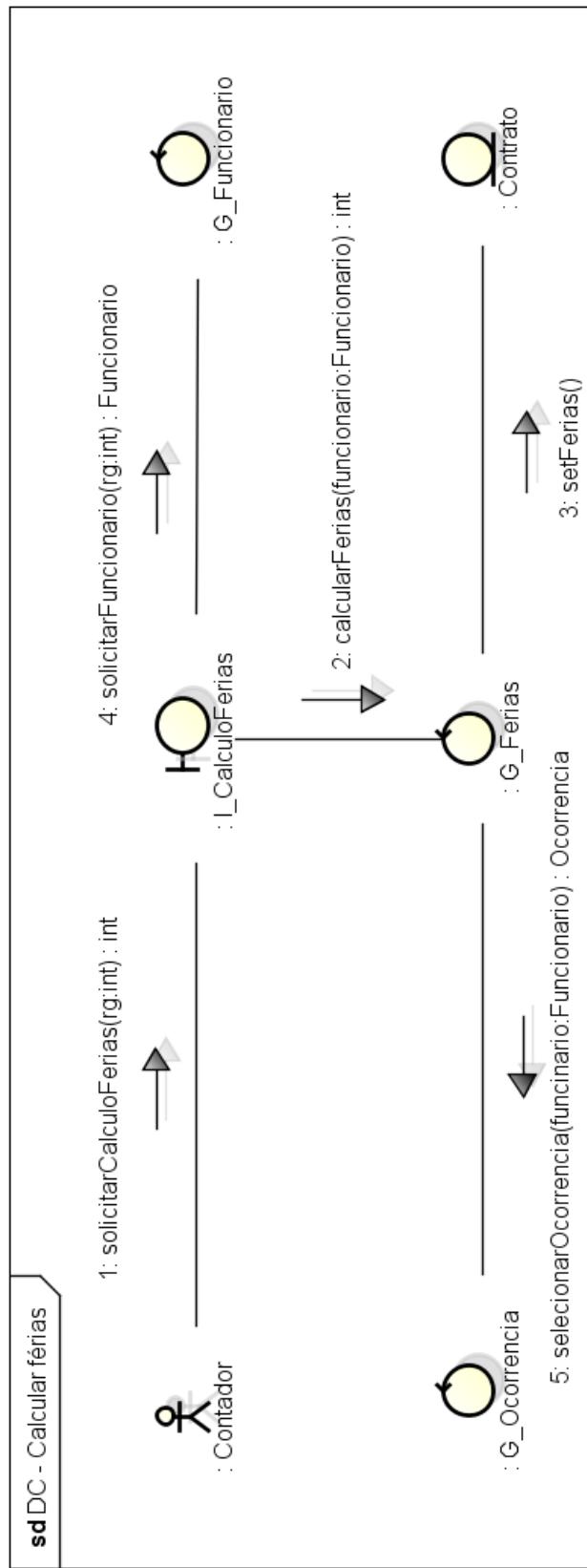


Figura 18 - Diagrama de Comunicação de Calcular férias

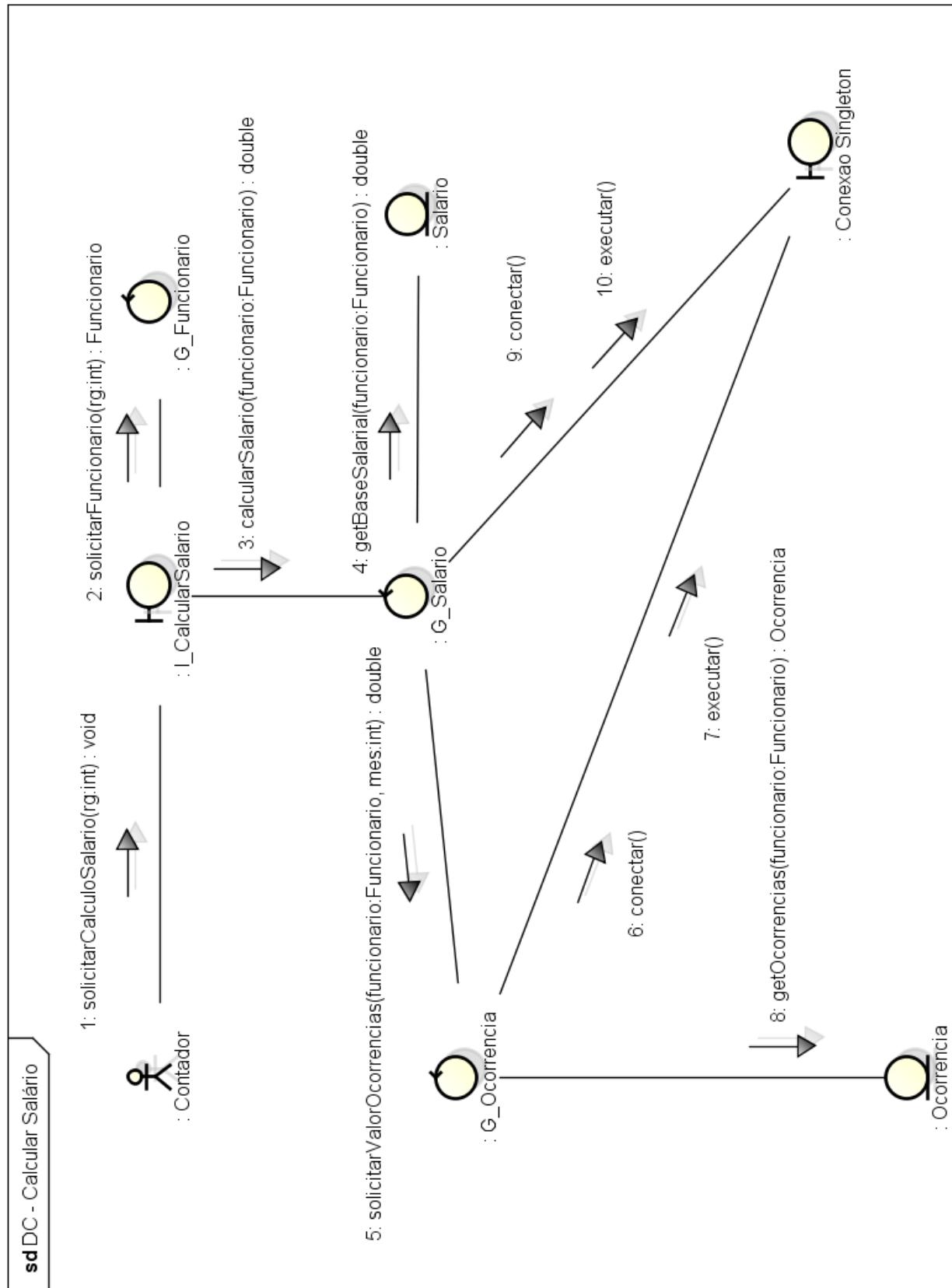


Figura 19 - Diagrama de Comunicação de Calcular salário

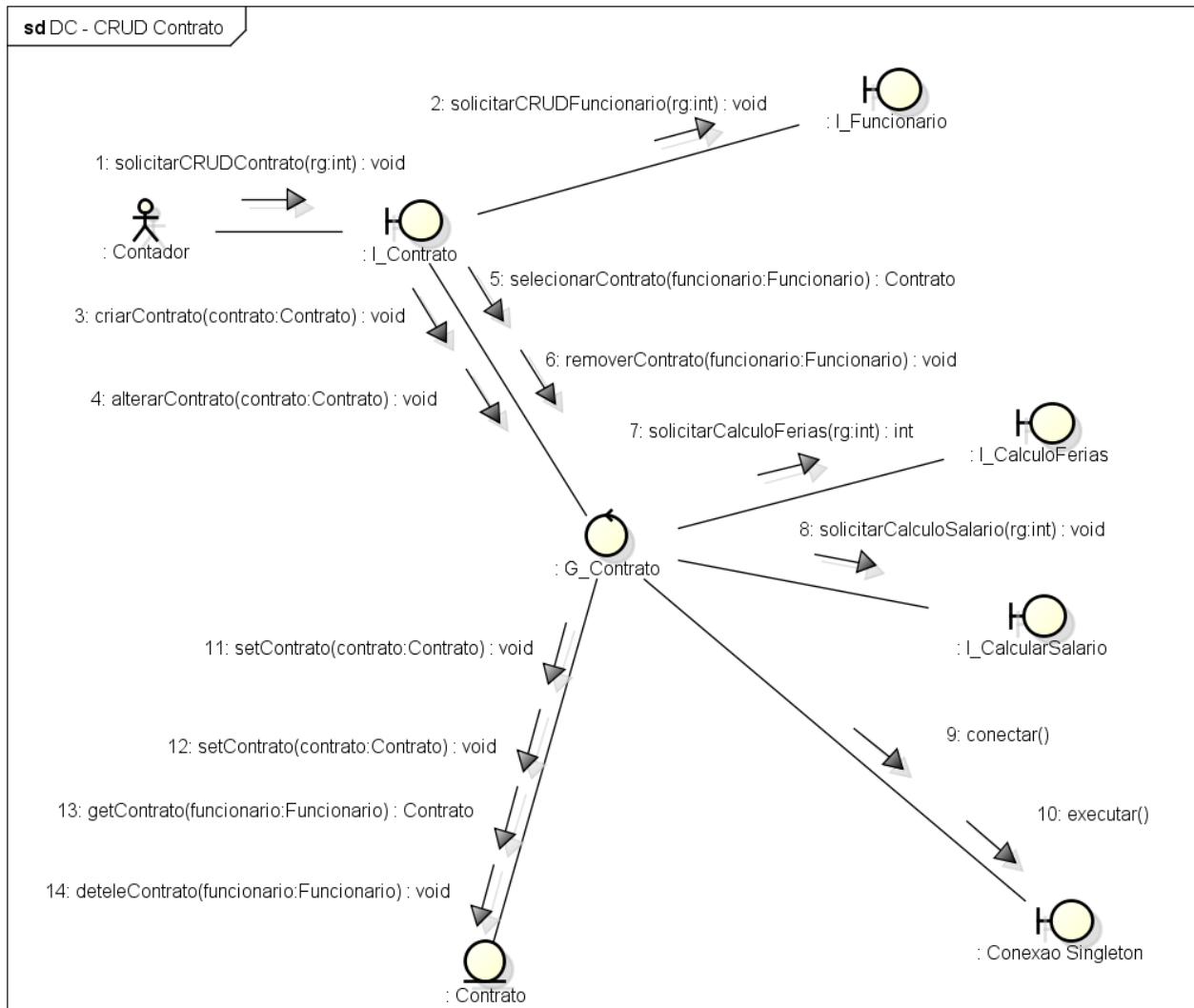


Figura 20 - Diagrama de Comunicação de CRUD Contrato

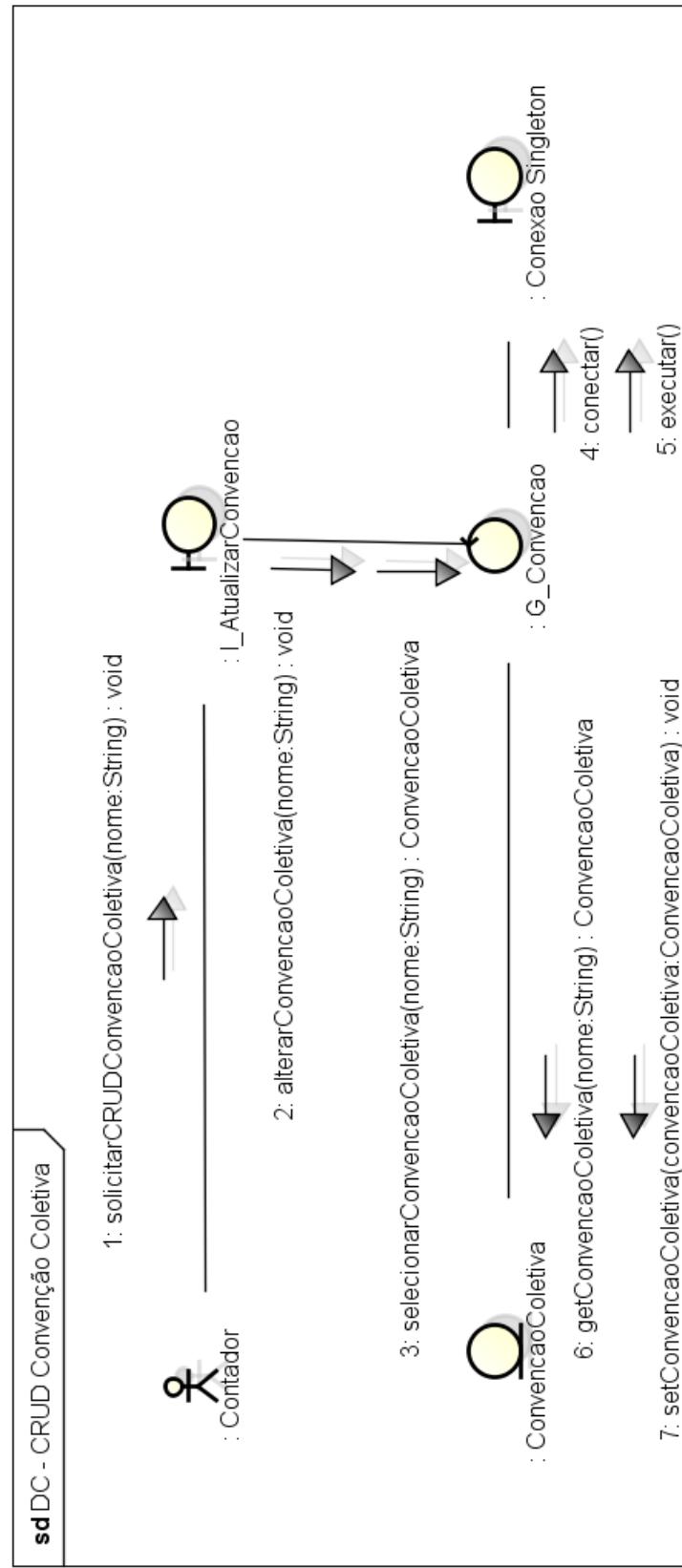


Figura 21 - Diagrama de Comunicação de CRUD Convenção coletiva

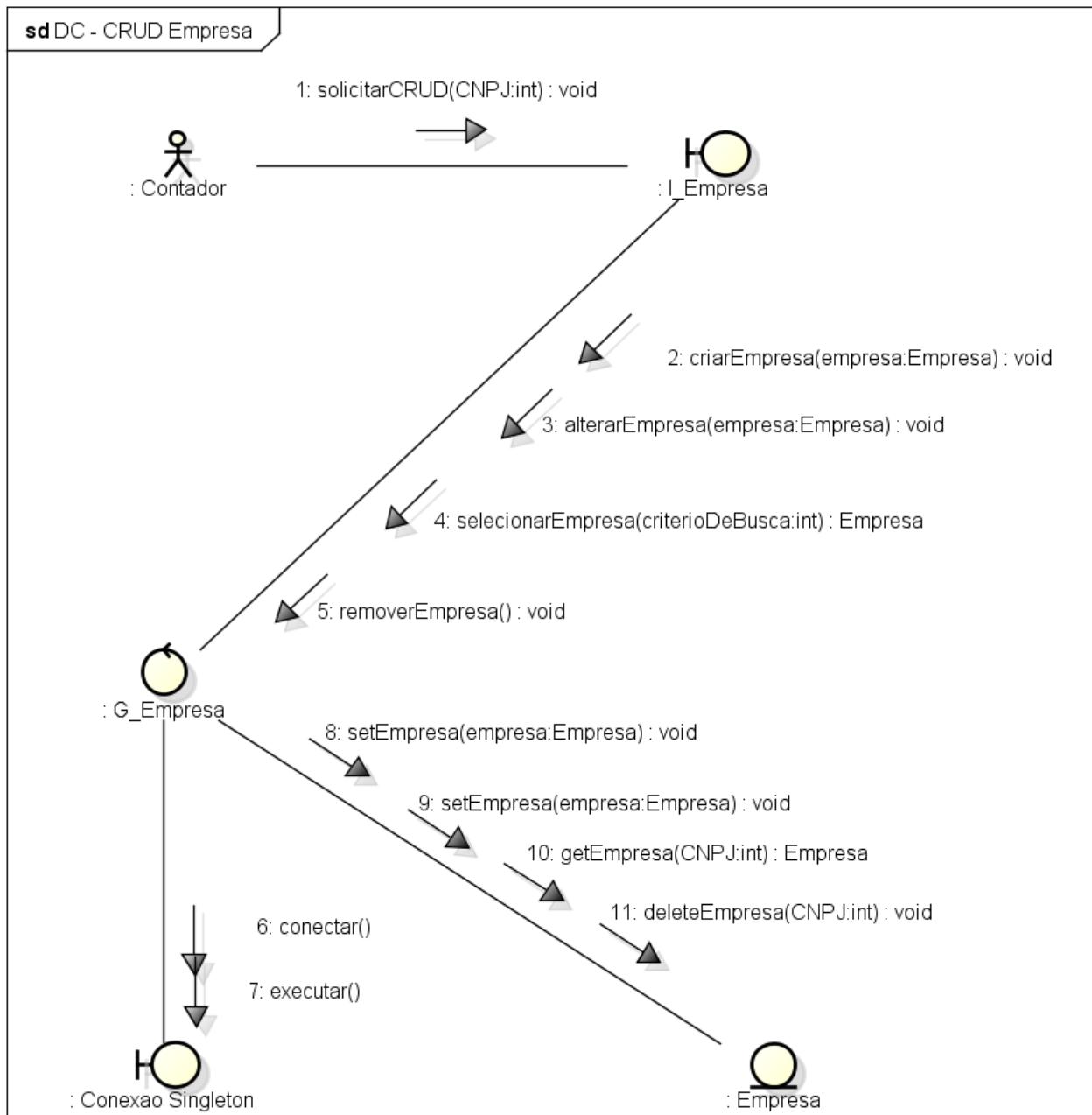


Figura 22 - Diagrama de Comunicação de CRUD Empresa

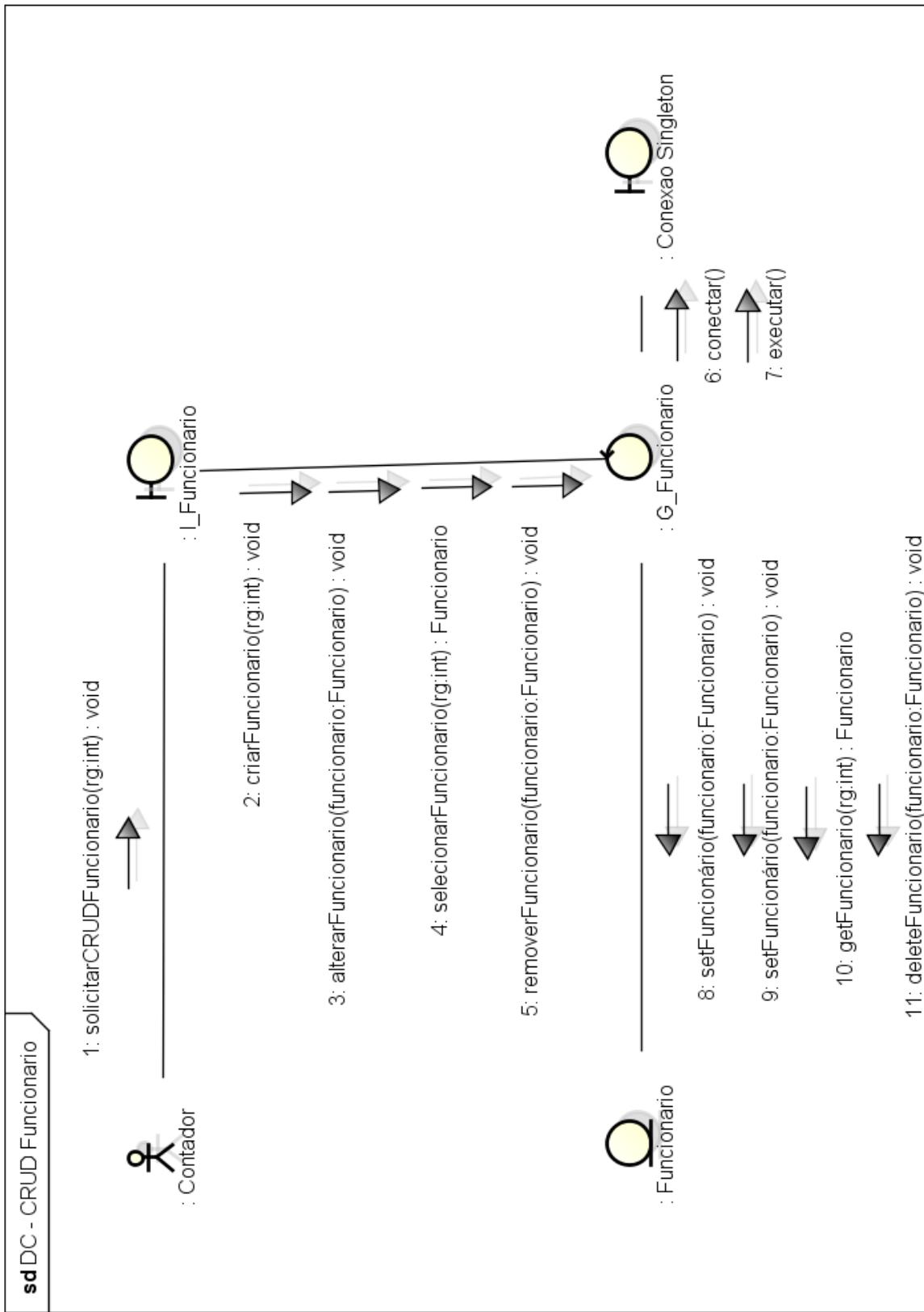


Figura 23 - Diagrama de Comunicação de CRUD Funcionário

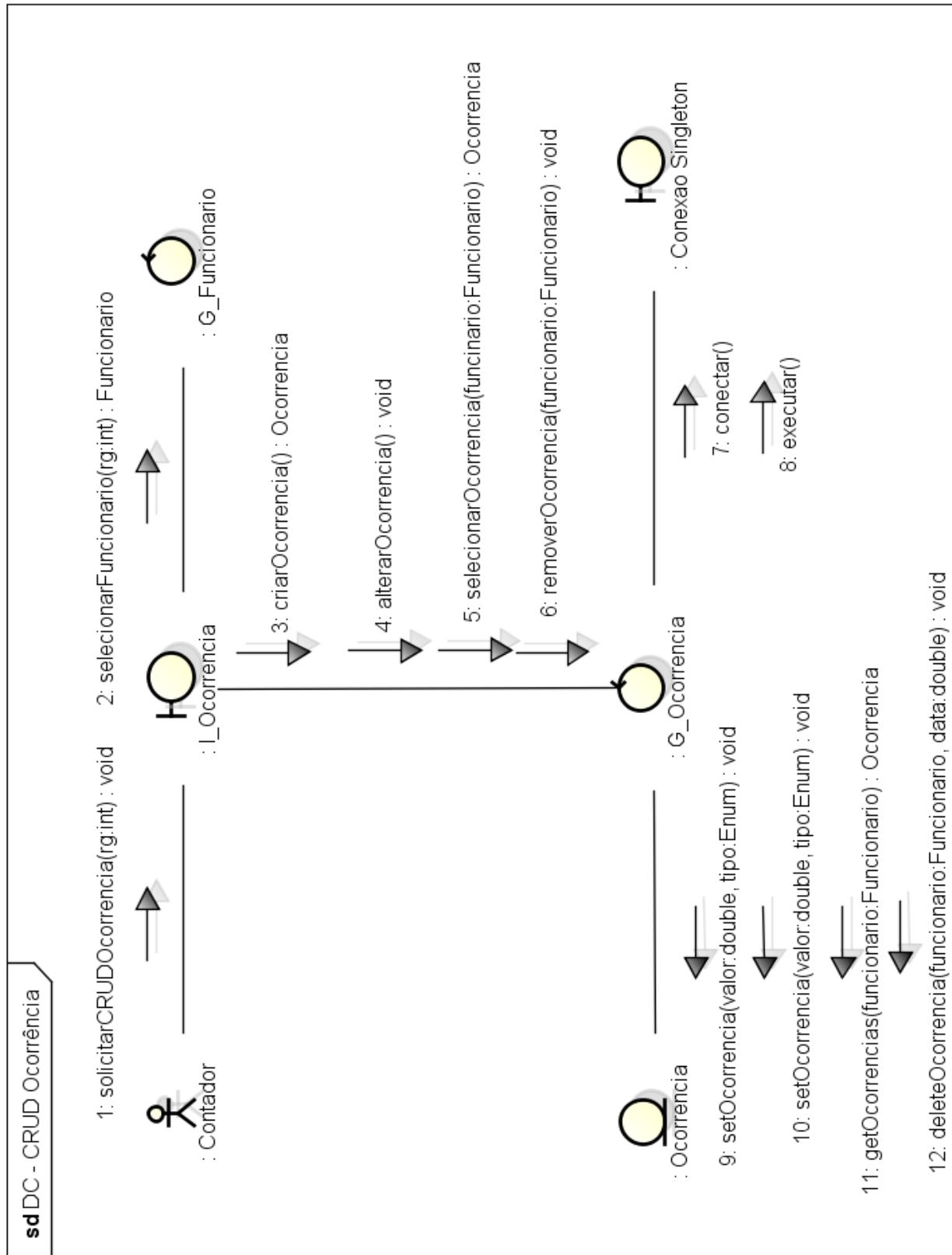


Figura 24 - Diagrama de Comunicação de CRUD Ocorrência

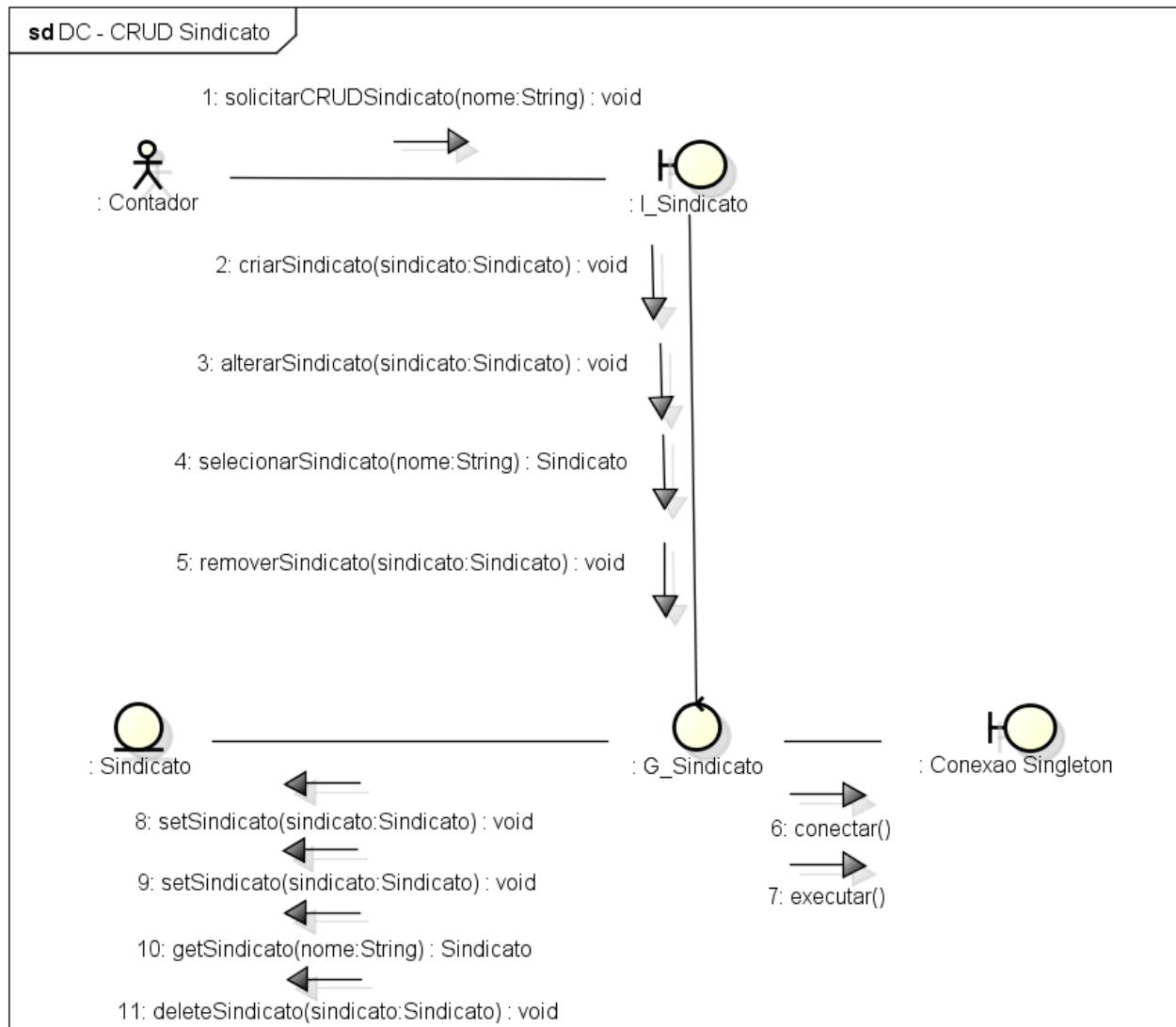


Figura 25 - Diagrama de Comunicação de CRUD Sindicato

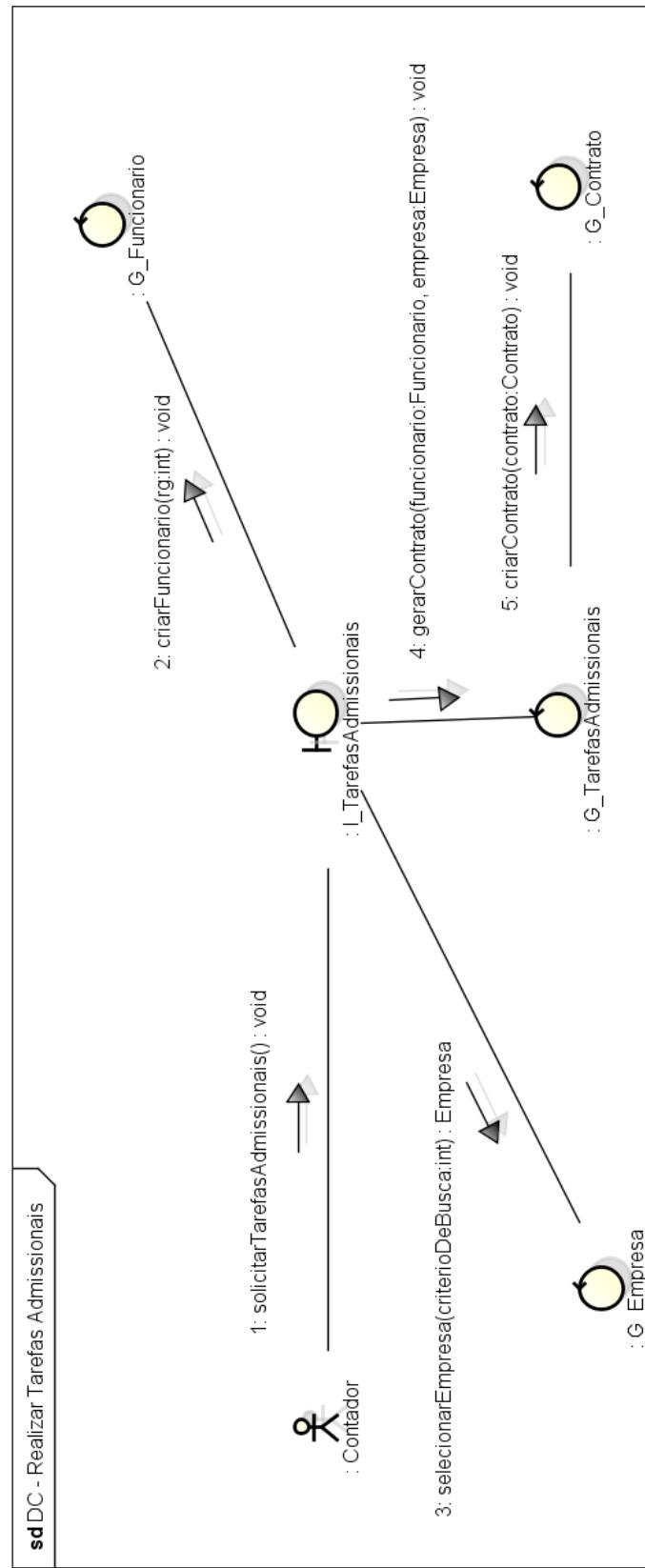


Figura 26 - Diagrama de Comunicação de Realizar tarefas admisionais

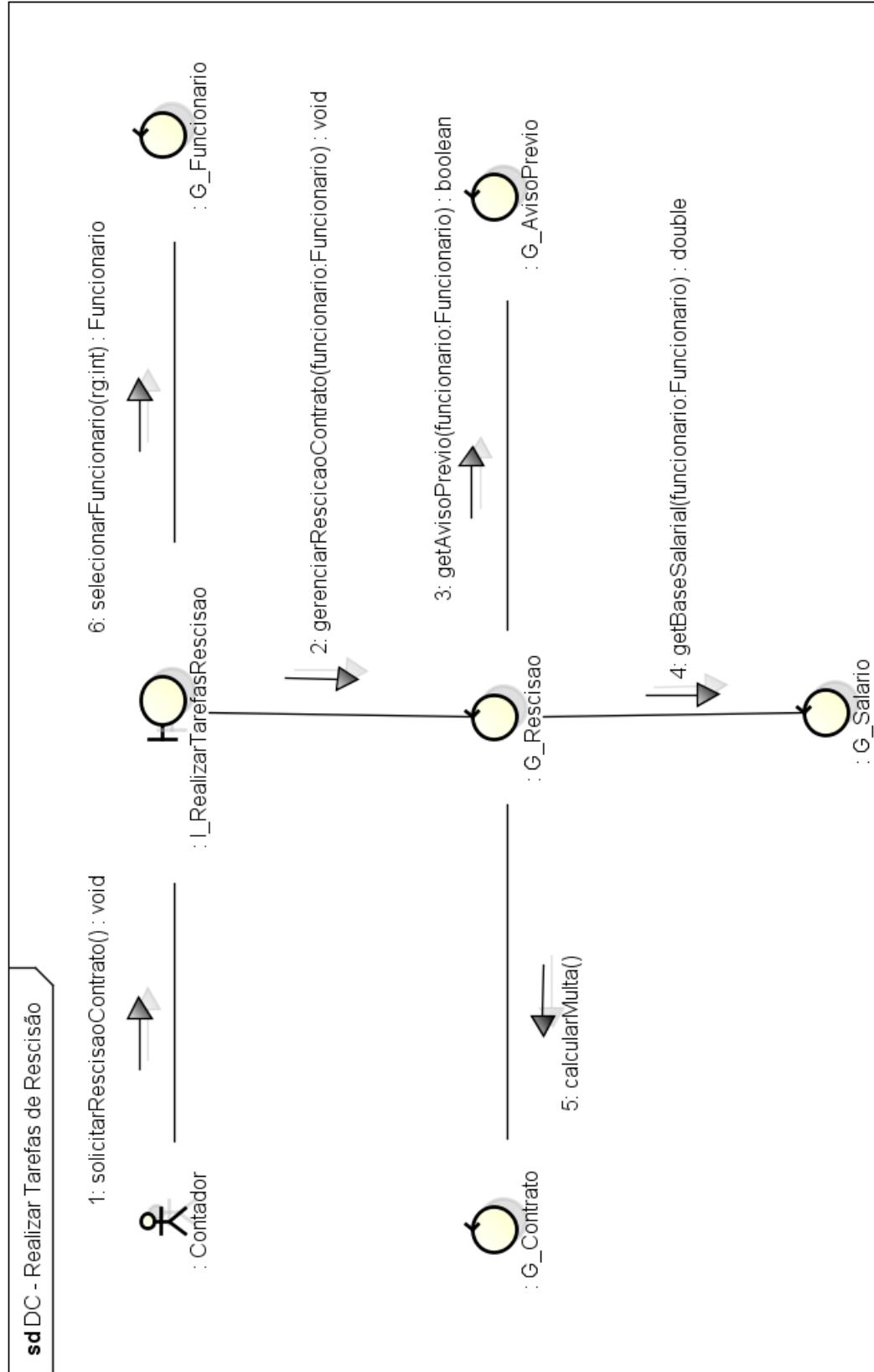


Figura 27 - Diagrama de Comunicação de Realizar tarefas de rescisão

10 Arquitetura de Software

Com base em discussões realizadas no *Software Engineering Institute da Carnegie Mellon University*, David Galan e Dewayne Perry definiram arquitetura de software como: “A estrutura dos componentes de um programa/sistema, seus inter-relacionamentos, princípios e diretrizes guiando o projeto e evolução ao longo do tempo”.

Em seguida, define-se o estilo arquitetural que será aplicado no sistema, bem como o padrão de projeto e o *framework*.

10.1 Estilos Arquiteturais

Os estilos de arquitetura expressam esquemas de organização estrutural de sistemas, fornecendo um conjunto de componentes do sistema, suas responsabilidades e a forma de interação entre eles, estabelecendo um padrão de utilização (é um padrão de organização de sistemas).

10.1.1 Estilo escolhido: Orientado a Objetos

O estilo arquitetural utilizado no sistema será o orientado a objetos, que tem como base o uso de um tipo de dados abstrato: o próprio objeto. Este estilo será utilizado pois objetos são entidades independentes que podem sofrer modificação, já que toda informação pertinente é mantida no mesmo. Por se tratar de um software de contabilidade com foco exclusivo no setor de recursos humanos, este sistema possui objetos específicos (tais como funcionário, empresa, salário, e assim por diante) e seus respectivos dados (por exemplo, para o objeto funcionário, têm-se nome, CPF, endereço, RG, entre outros).

Além disso, objetos possuem mecanismos de trocas de mensagens, tornando possível a requisição ou oferta de um objeto para outro (exemplificando, o objeto salário requisita dados do objeto ocorrência para que o salário final seja calculado). Outro fator decisivo foi a ocultação da informação (a base desse estilo arquitetural). Tratando-se de um sistema de recursos humanos, certas informações de um objeto são irrelevantes ou mesmo privadas para outros objetos.

10.1.2 Estilos rejeitados: justificativa

O estilo *Pipes e Filtros* considera a existência de uma rede pela qual flui dados de uma extremidade à outra. O fluxo de dados se dá através de *pipes* e os dados se transformam quando processados nos filtros. Como o compartilhamento de informações é caro e inflexível, ocorrem *overheads* nas transformações dos dados, o tratamento de erros é difícil de ser obtido, e faltam vantagens relevantes para o uso deste estilo neste sistema.

O estilo *Camadas* auxilia na aplicação de estruturas que podem ser decompostas em grupos de subtarefas pertencentes a um nível particular de abstração. Este estilo foi rejeitado, pois possui baixa eficiência e causa impacto no desempenho, além de não possuir vantagens relevantes ou soluções adequadas para os requisitos funcionais e não funcionais deste sistema.

O estilo *Invocação Implícita* idealiza componentes que interagem indiretamente e implicitamente, em resposta a uma notificação ou evento. Apesar de parecer com um estilo necessário neste sistema, o fato dos componentes abrirem mão do controle sobre o tipo de computação ou ordem de processamento a serem realizados, e a não interferência dos próprios componentes na ordem de invocação dos eventos são algumas desvantagens a se considerar. Mesmo com os seus benefícios, não é preciso utilizar a *Invocação Implícita*, uma vez que o estilo Orientado a Objetos já satisfaz o necessário pelo sistema.

O estilo Orientado a Eventos possui componentes independentes que se comunicam somente por meio de eventos transmitidos por um barramento. Por necessitar de um grande overhead de processamento, o uso deste estilo torna-se inadequado para o sistema.

O estilo Quadro-Negro possui uma central de armazenamento (chamada de “Quadro-Negro”) de dados e vocabulário, e um componente de controle que monitora as mudanças nesta central e decide qual ação deve ser tomada para resolver os possíveis problemas. Como ele tem baixa eficiência, não possui solução de problemas garantido, precisa de um alto esforço de desenvolvimento e não seria útil em um sistema de contabilidade que também depende das ações de um usuário (o contador, no caso).

O estilo Orientado a Serviços permite que organizações tomem suas informações disponíveis a outros programas por meio da utilização de uma interface *web service*. Este sistema não é um provedor de serviços e não precisa tornar suas informações disponíveis.

O estilo Arquitetura em Nuvem possibilita o acesso a um conjunto de recursos computacionais configuráveis sob demanda e redimensionável. Possui recursos convenientes a empresas, universidade ou departamentos governamentais, tendo em vista possíveis usuários de diferentes interesses e/ou localizações. O sistema em questão possuirá apenas um usuário, o contador, logo, não precisa ser feito uma infraestrutura visando o uso de uma comunidade de usuários. Consequentemente, não haveria aumento da dependência de acesso a internet e nem custos relacionados a servidores e criptografia de dados.

10.2 Padrões de Projeto

Padrões de projeto são “ideias” de soluções para problemas já encontrados no desenvolvimento de software. Cada um possui uma abordagem e solução comprovada.

O padrão de projeto escolhido para ser utilizado no sistema foi o Singleton, uma vez que tal padrão é capaz de garantir a existência de uma única classe que possui acesso global ao seu objeto. Dessa maneira, evita-se duplicações de registros e outras complexidades desnecessárias, favorecendo sistemas que compõem diversos componentes que acessam apenas uma instância de classe. No caso, o acesso ao banco de dados, recurso fundamental para o sistema contábil.

10.3 Frameworks

Frameworks são estruturas genéricas que podem ser estendidas para criar aplicações ou subsistemas mais específicos. Em outras palavras, são códigos prontos que tratam uma abordagem de maneira geral a fim de serem utilizadas de maneira mais específica.

O *framework* a ser utilizado será o MySQL++, uma vez que a linguagem de programação escolhida para o desenvolvimento do sistema é o C++, e pelo fato deste *framework* servir como interface para a comunicação com servidores MySQL (o SGBD a ser utilizado). Serve também para desenvolvimento de aplicações para banco de dados, além de uma maior quantidade de exemplos e tutoriais disponíveis.

Outro *framework* a ser utilizado é o gtkmm, um *software* livre distribuído sob a Licença Pública Geral de Biblioteca GNU (GPL). O gtkmm é uma interface oficial para C++ para a interface gráfica GTK+. Um de seus destaques é que incluem chamadas seguras e um conjunto de widgets de fácil extensão através de herança. Com o gtkmm é possível criar interfaces via código ou, neste caso, com o construtor de interfaces Glade, usando o Gtk::Builder.

10.4 Alterações no projeto

Devido à má documentação do MySQL++, foi decidido alterar a linguagem e, consequentemente, o *framework* e a ferramenta para a construção de interfaces. Agora, trabalharemos com a linguagem Java, com o *framework* *Hibernate* e para a construção das interfaces o *widget toolkit GUI*. O NetBeans é a IDE utilizada.

11 Diagramas de Sequência

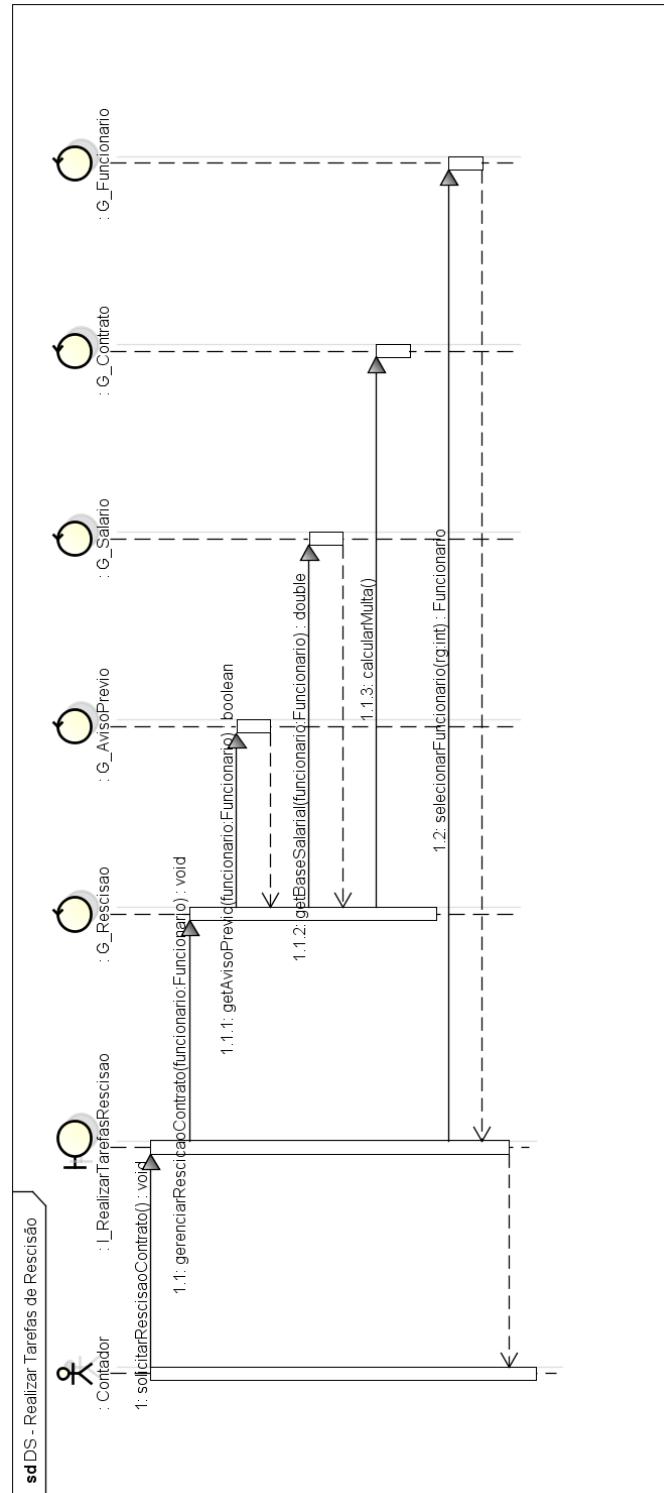


Figura 28 – Diagrama de Sequência de Realizar tarefas de rescisão (Christian)

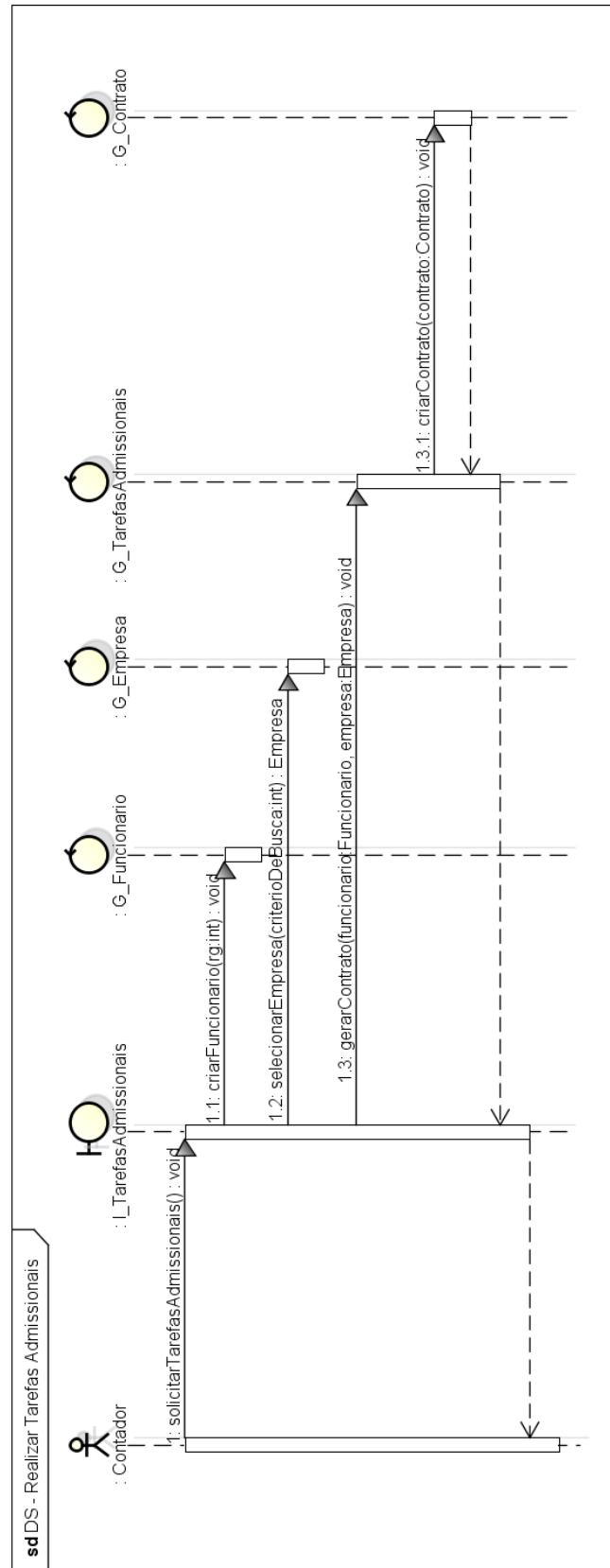


Figura 29 – Diagrama de Sequência de Realizar tarefas de admissão (Christian)

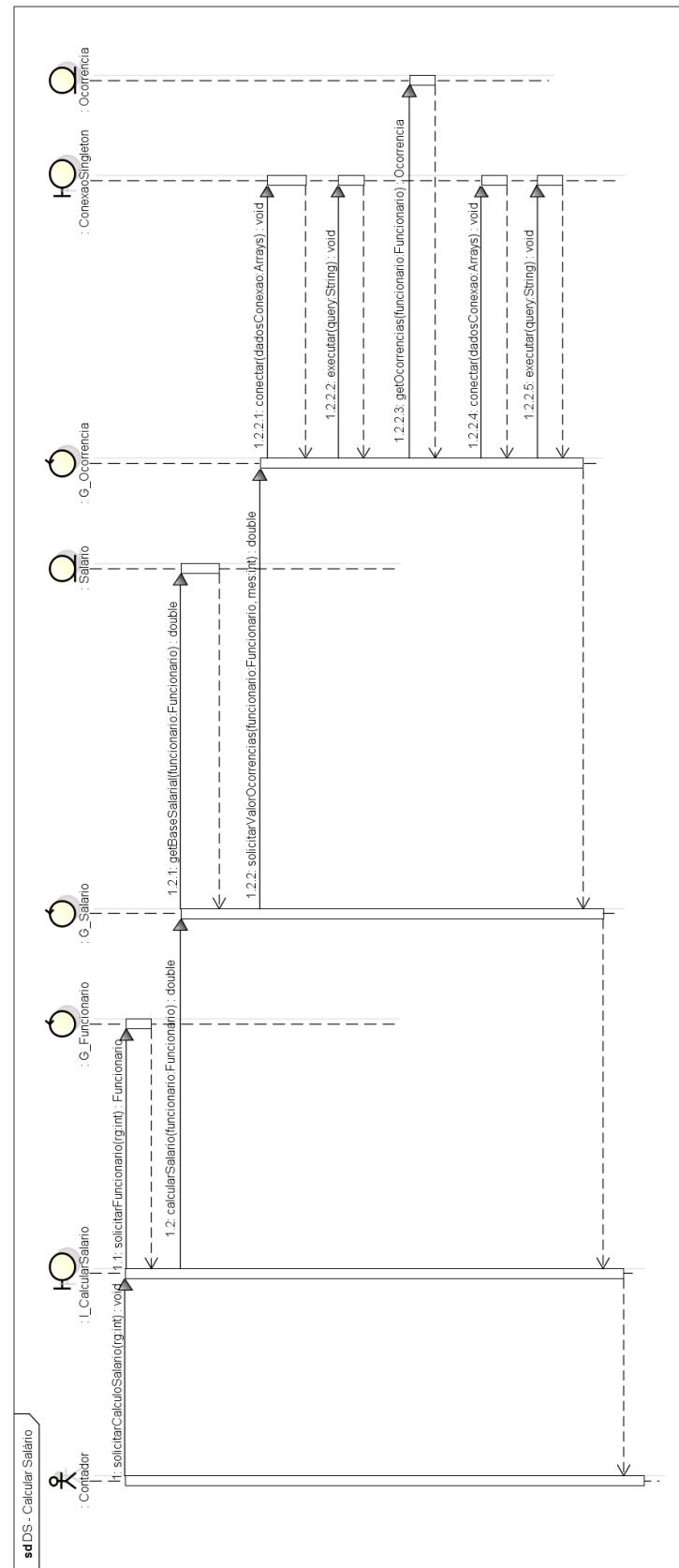


Figura 30 – Diagrama de Sequência de Calcular salário (Kevin)

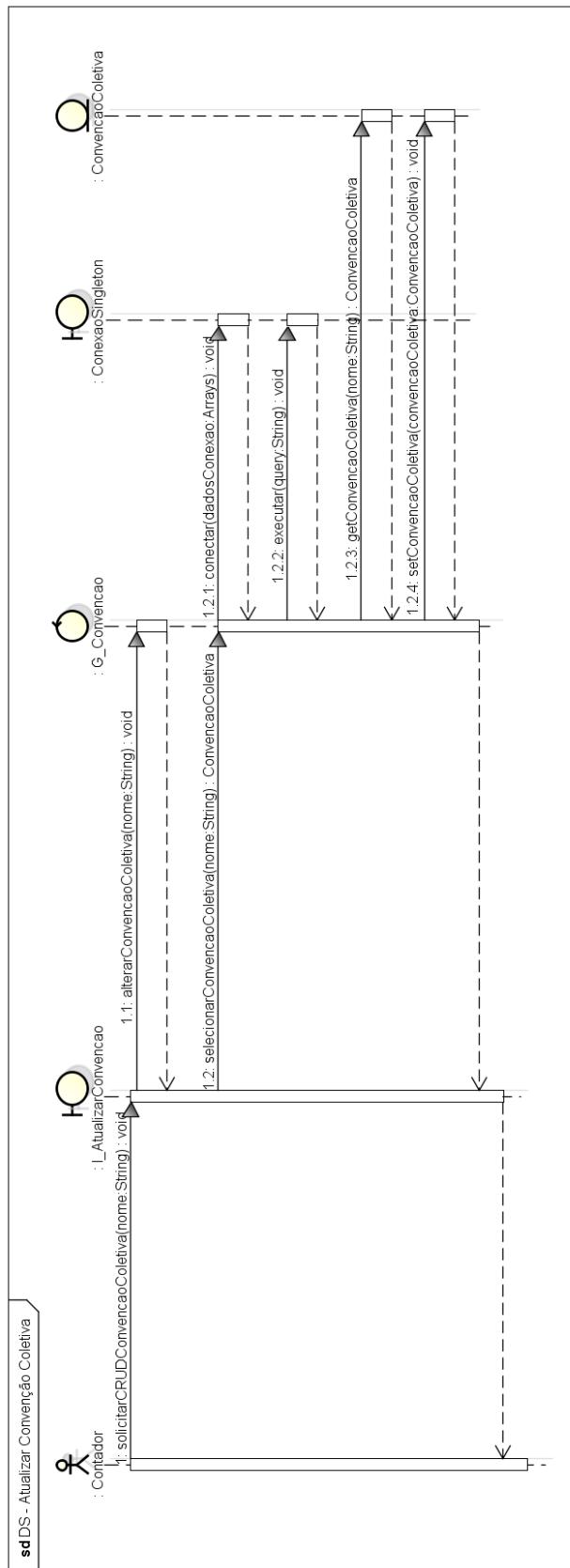


Figura 31 – Diagrama de Sequência de Atualizar Convenção Coletiva (Kevin)

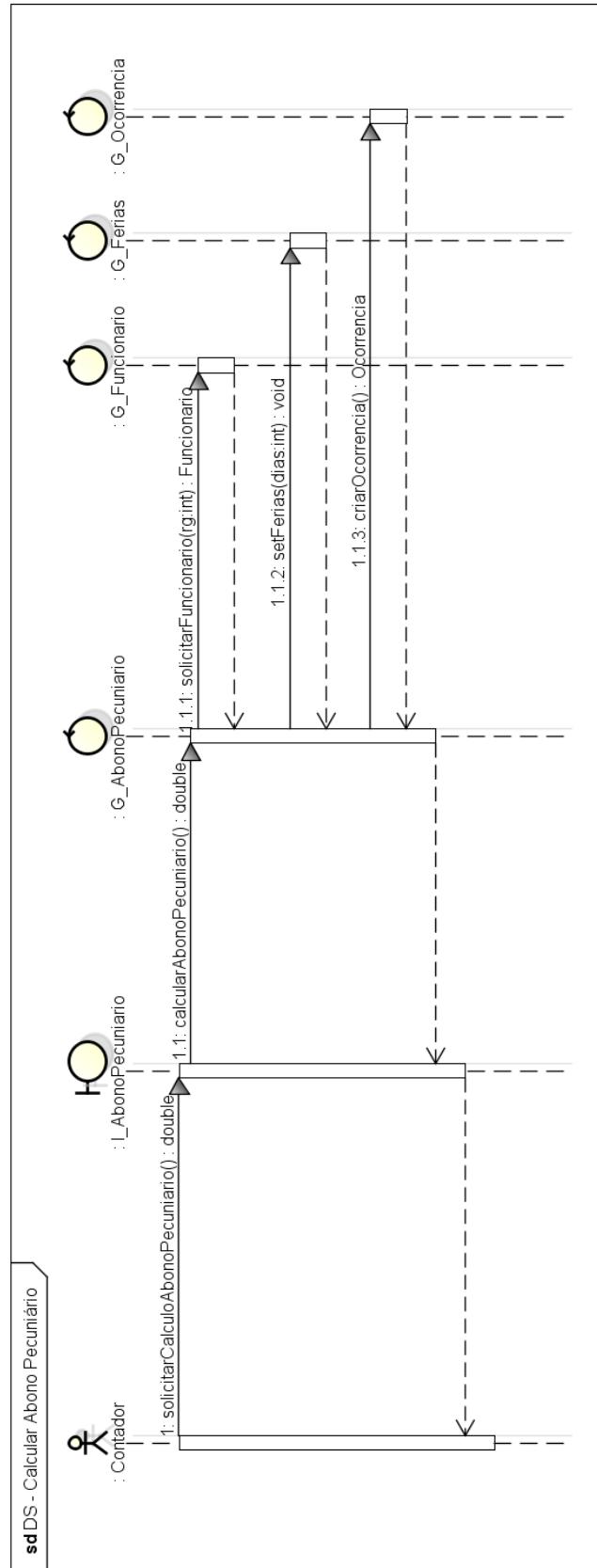


Figura 32 – Diagrama de Sequência de Calcular Abono pecuniário (Alessandra)

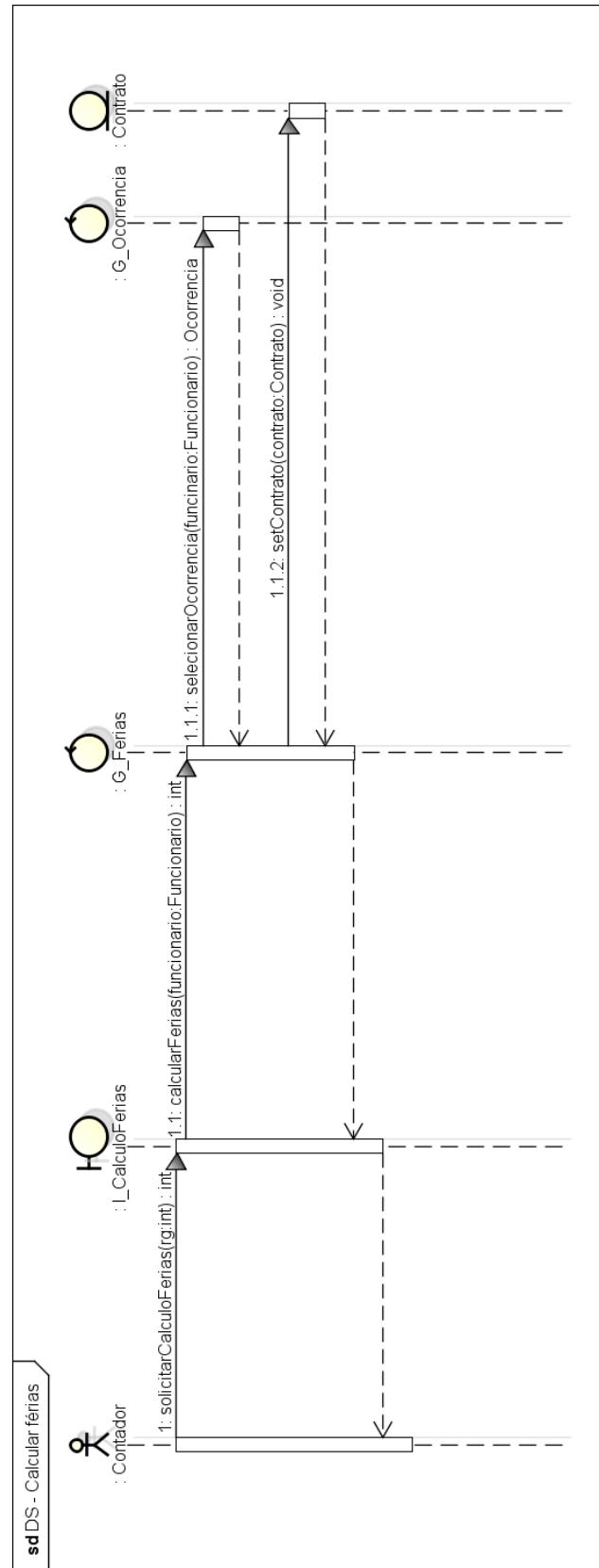


Figura 33 – Diagrama de Sequência de Calcular férias (Alessandra)

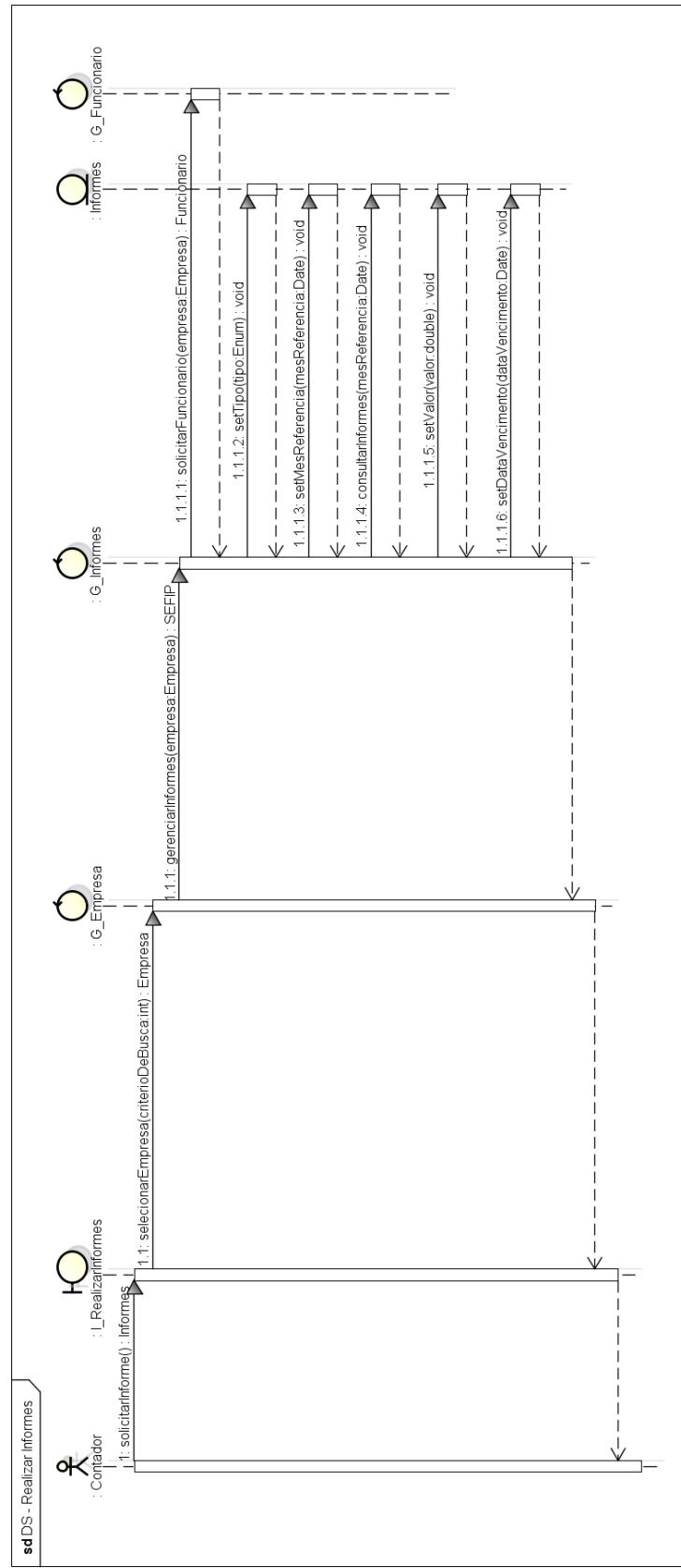


Figura 34 – Diagrama de Sequência de Realizar informes (Pedro)

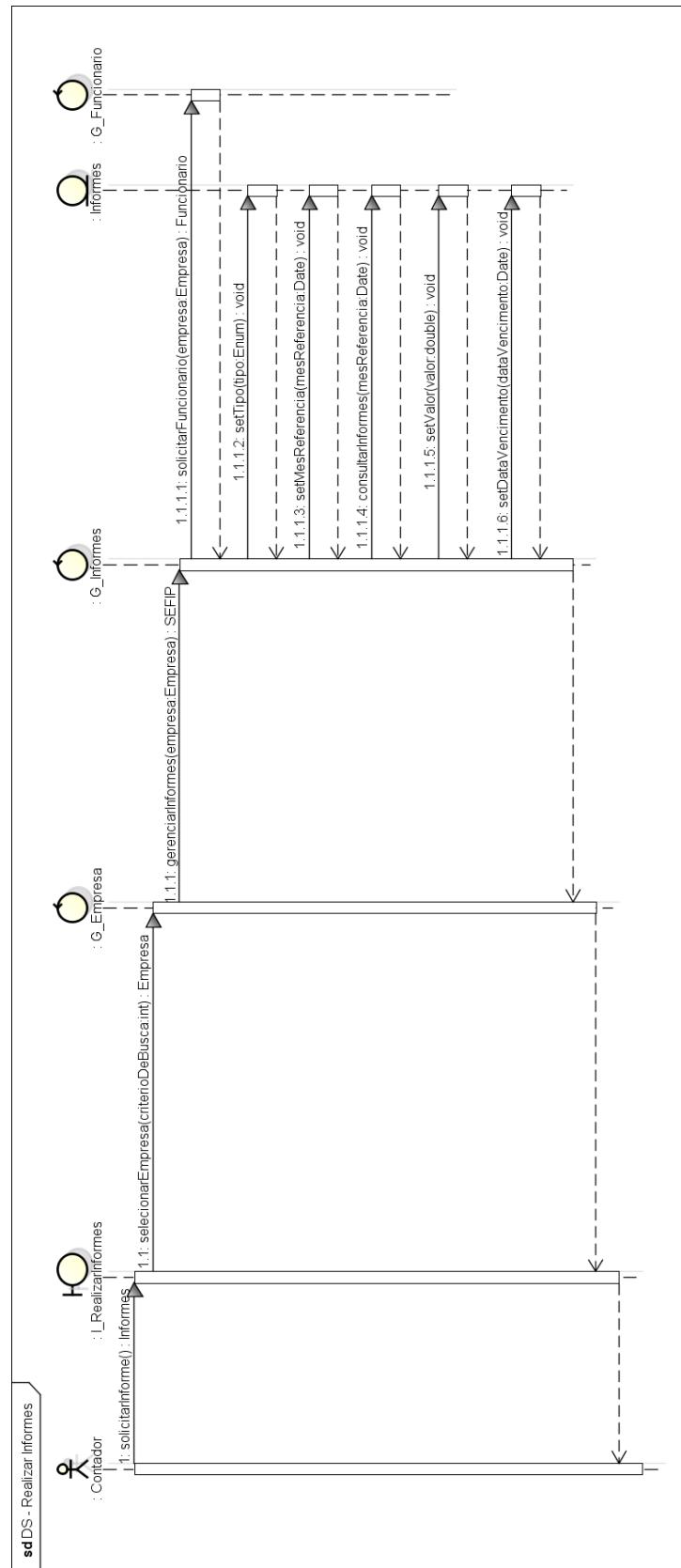


Figura 35 – Diagrama de Sequência de CRUD Ocorrência (Pedro)

12 Diagramas de Estados

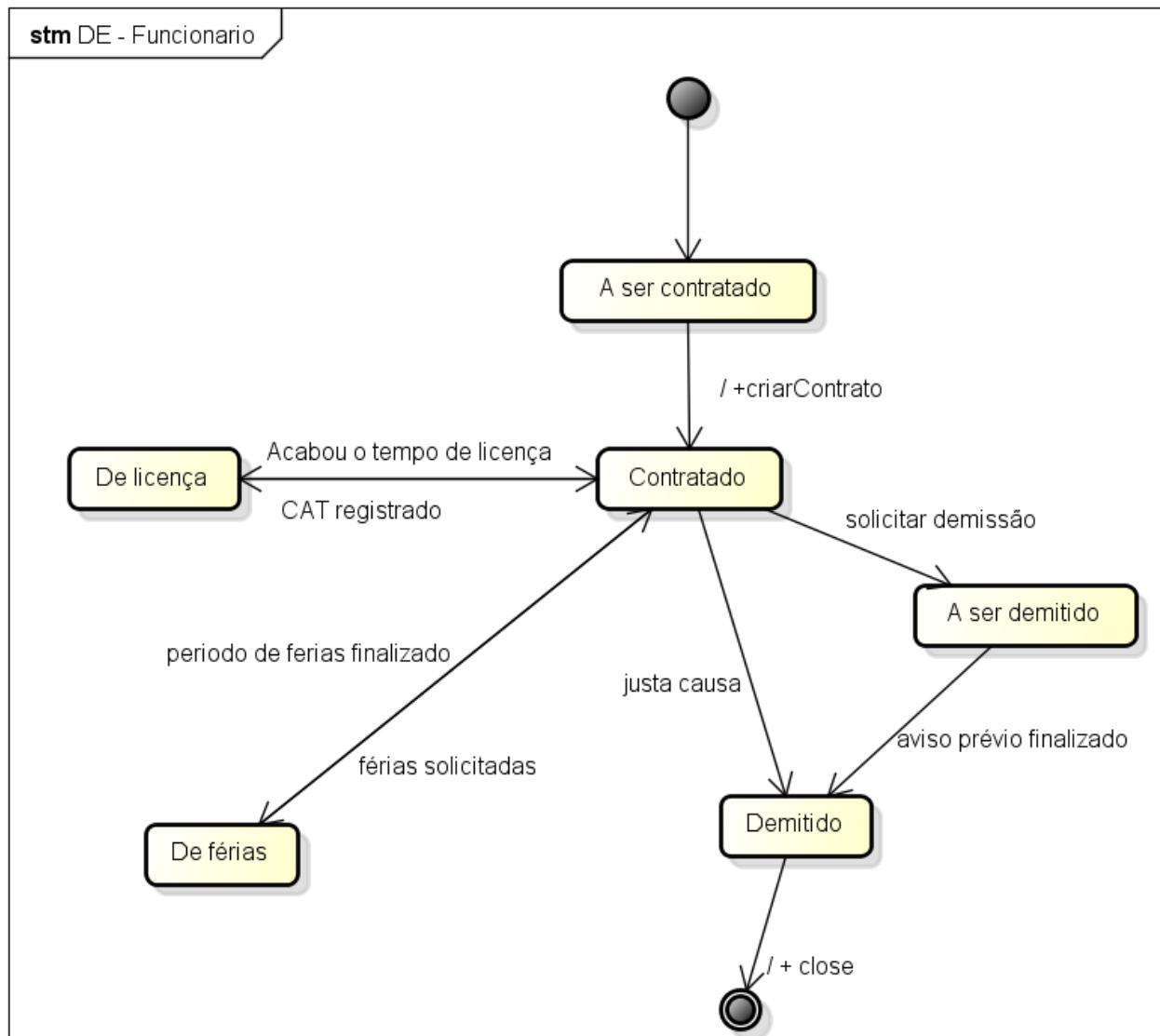


Figura 36 – Diagrama de Estados de Funcionário (Pedro)

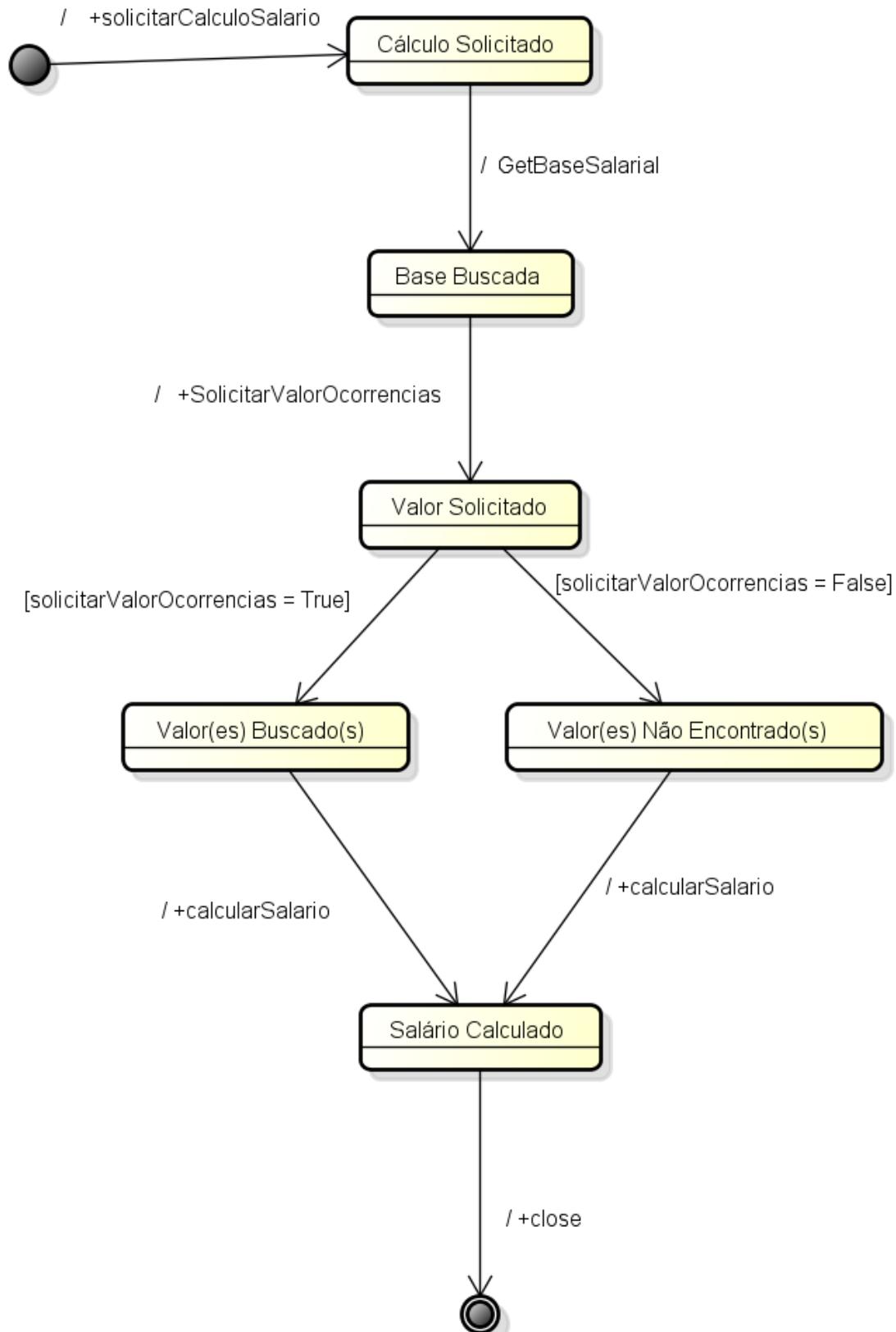


Figura 37 – Diagrama de Estados de Salário (Kevin)

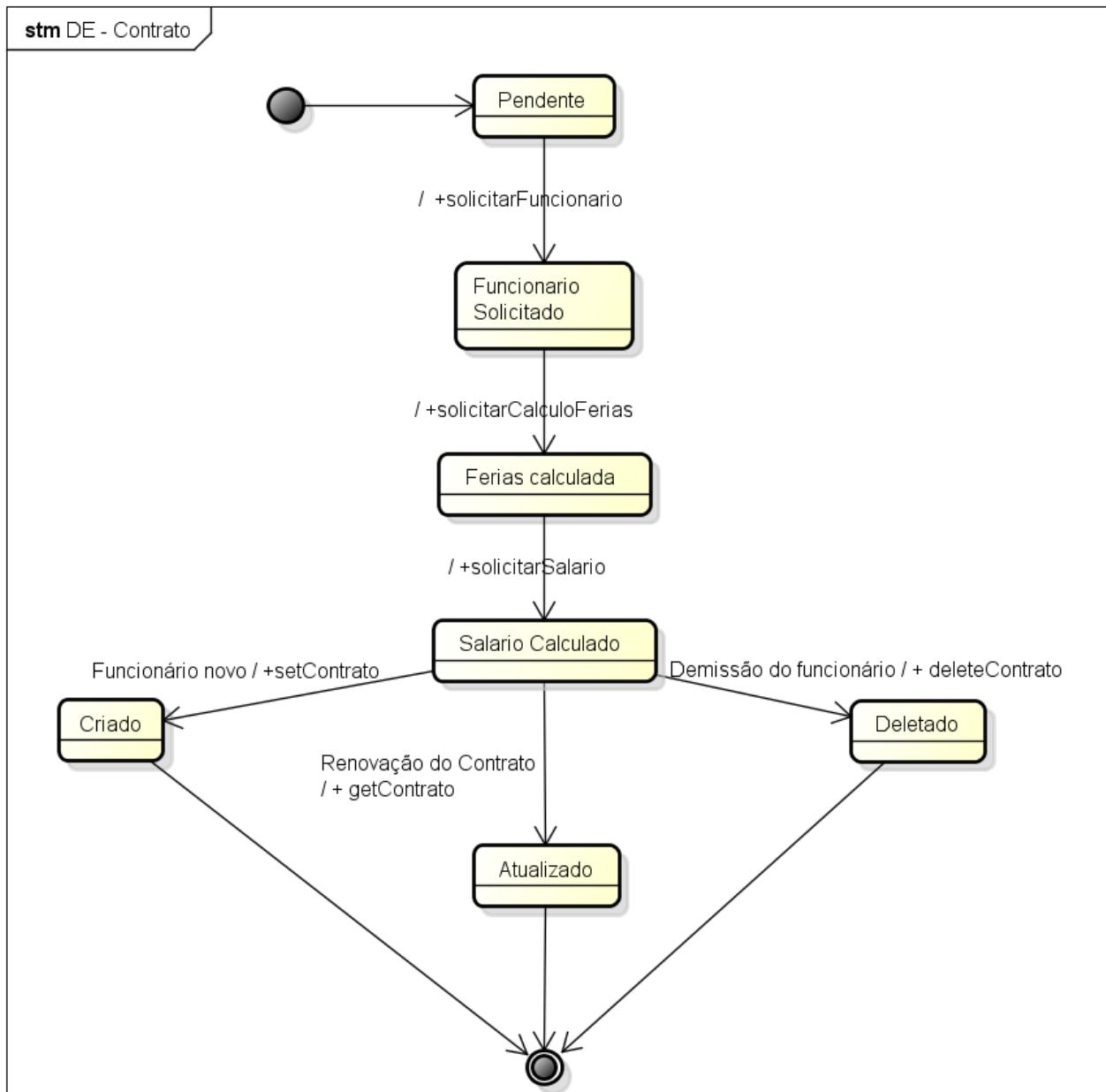


Figura 38 – Diagrama de Estados de Contrato (Christian)

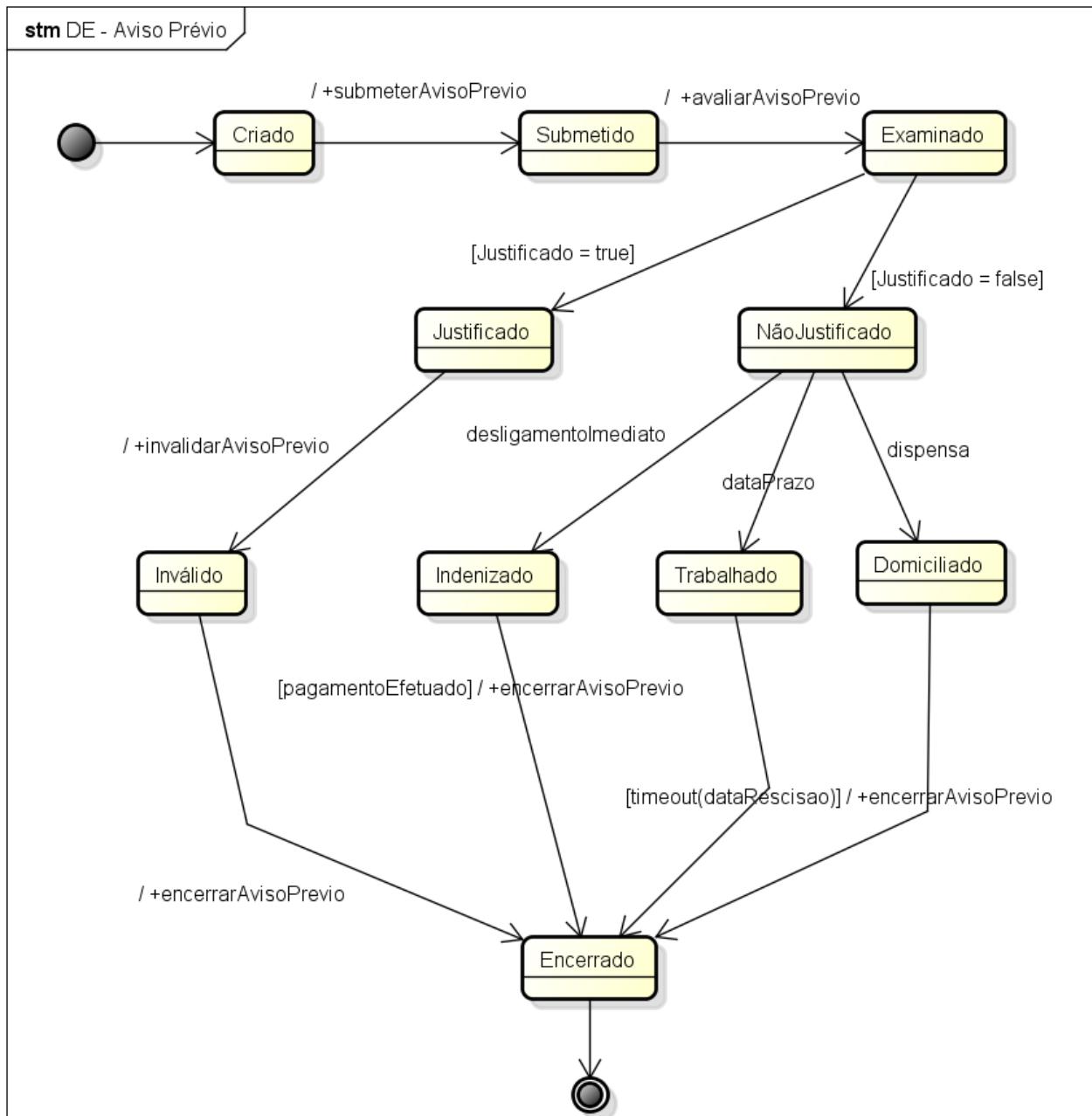


Figura 39 – Diagrama de Estados de Aviso prévio (Alessandra)

13 Diagrama de Atividades

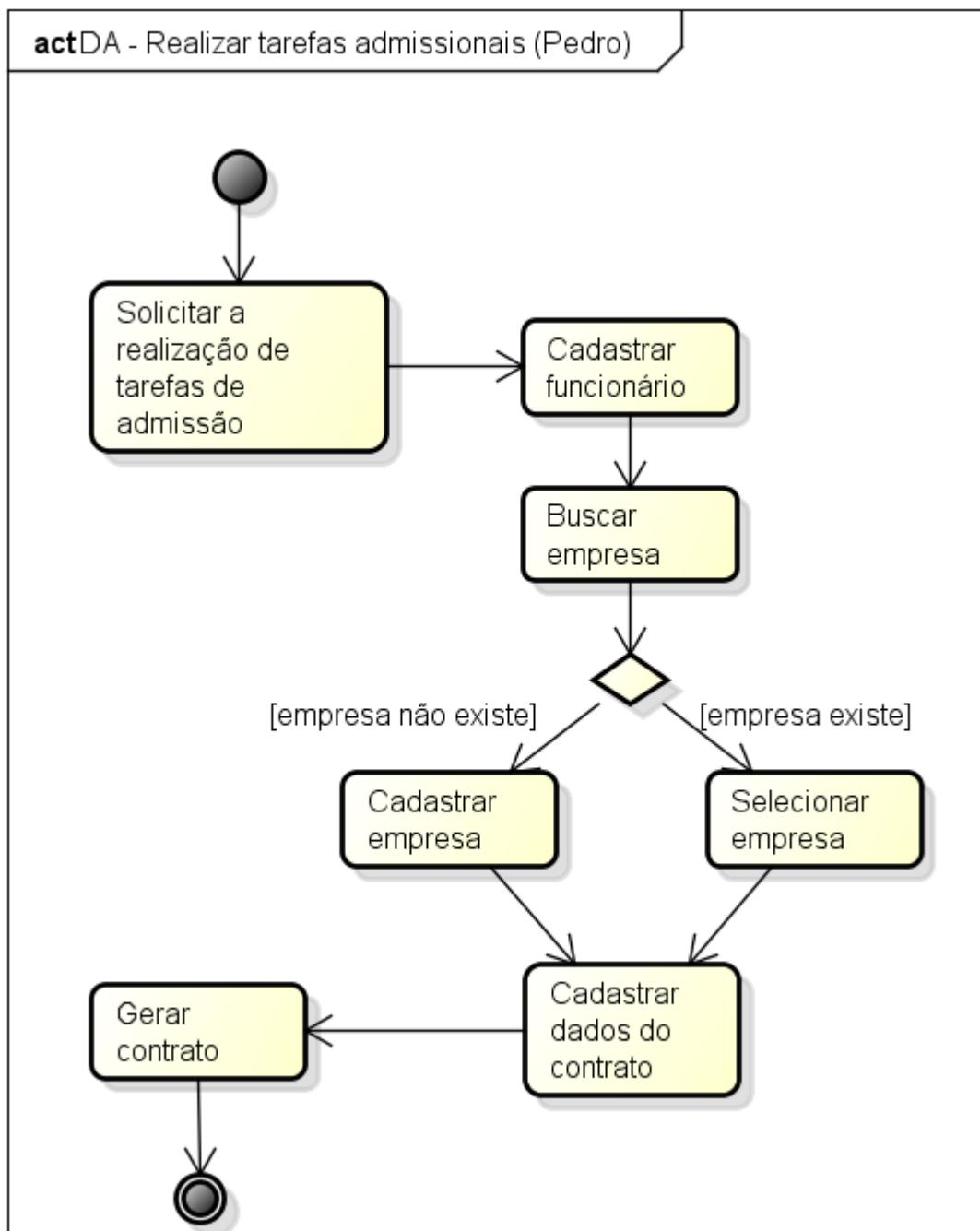


Figura 40 – Diagrama de Atividades Realizar tarefas admissionais (Pedro)

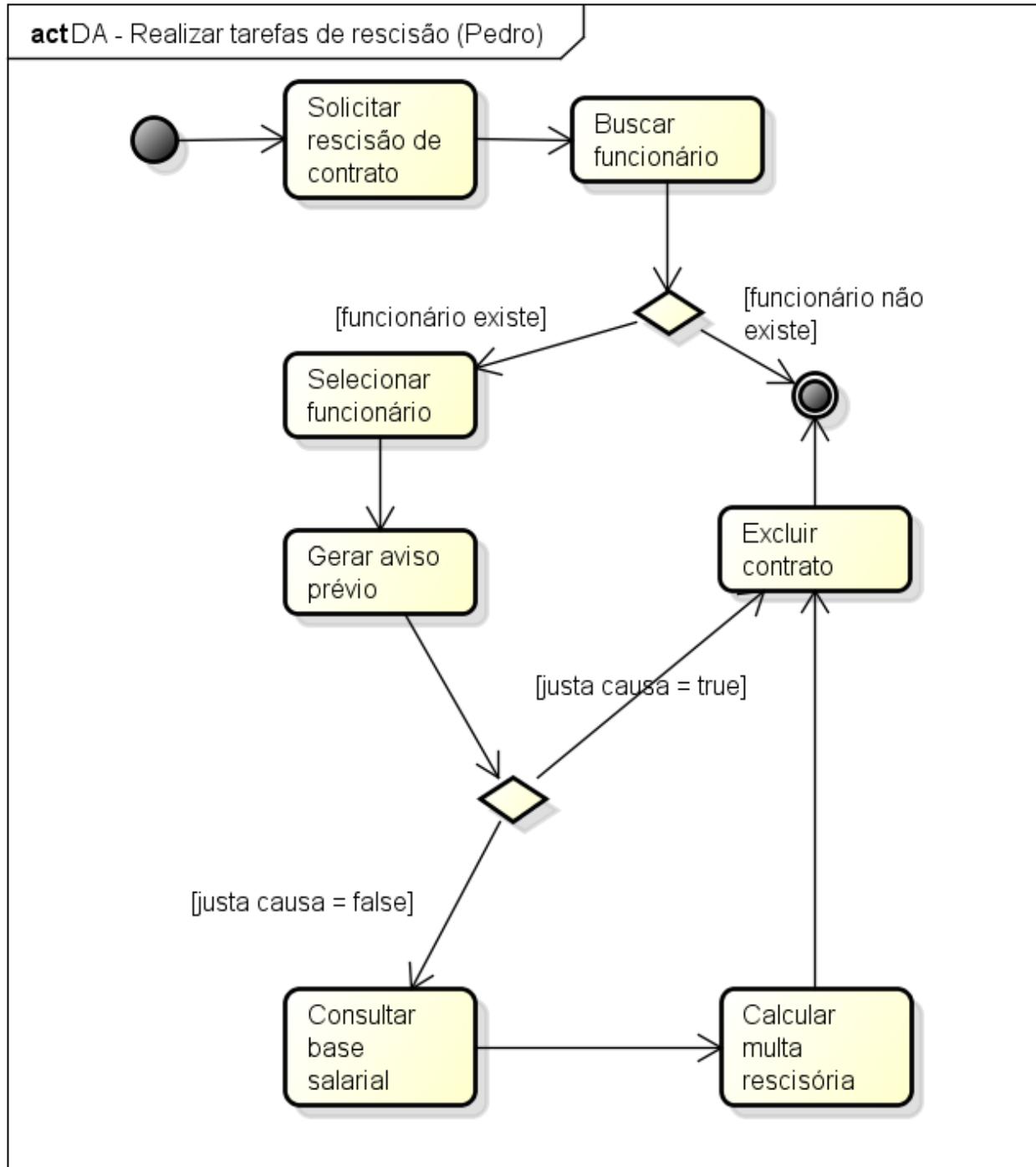


Figura 41 – Diagrama de Atividades Realizar tarefas de rescisão (Pedro)

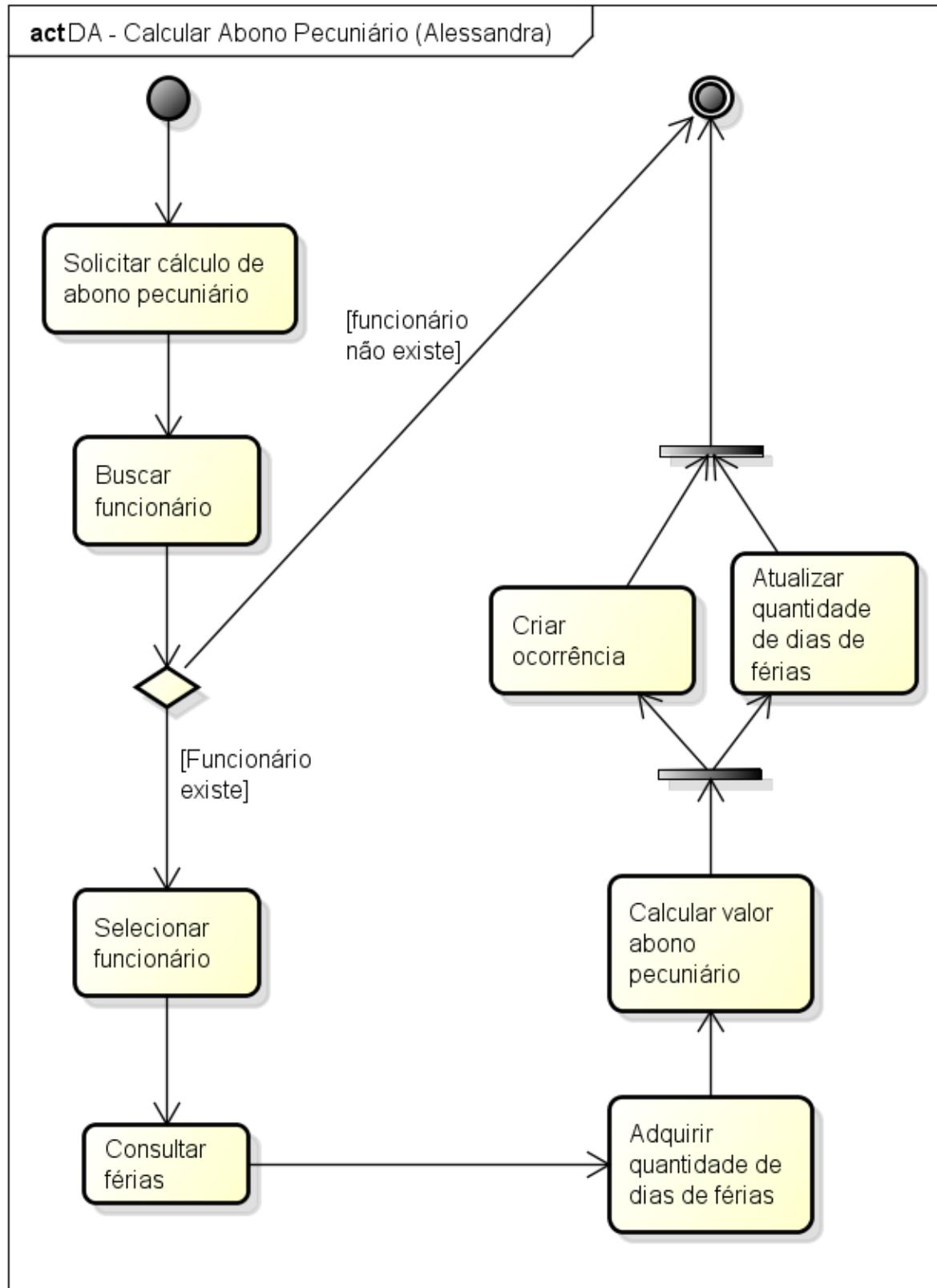


Figura 42 – Diagrama de Atividades de Calcular Abono pecuniário (Alessandra)

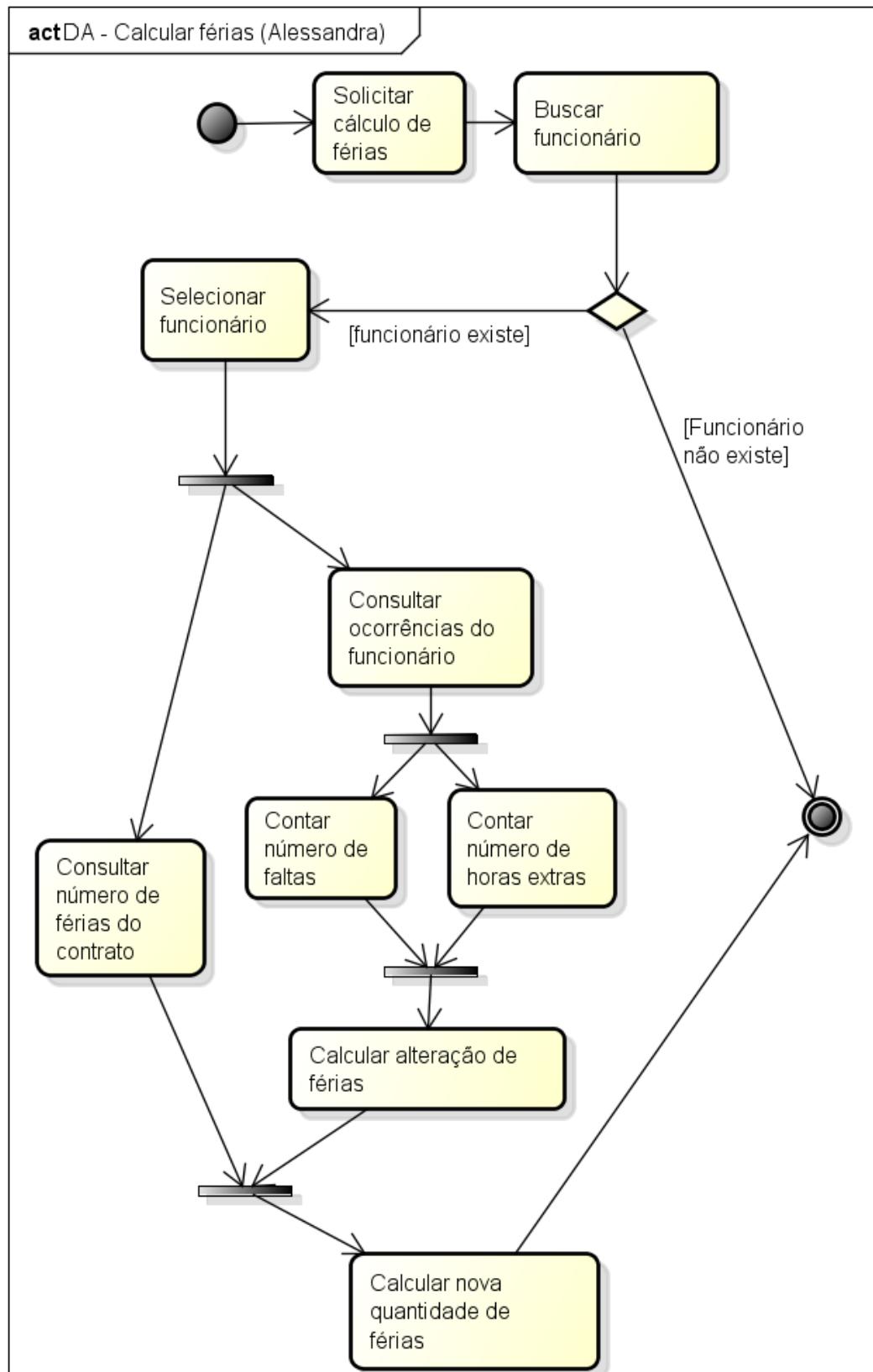


Figura 43 – Diagrama de Atividades de Calcular férias (Alessandra)

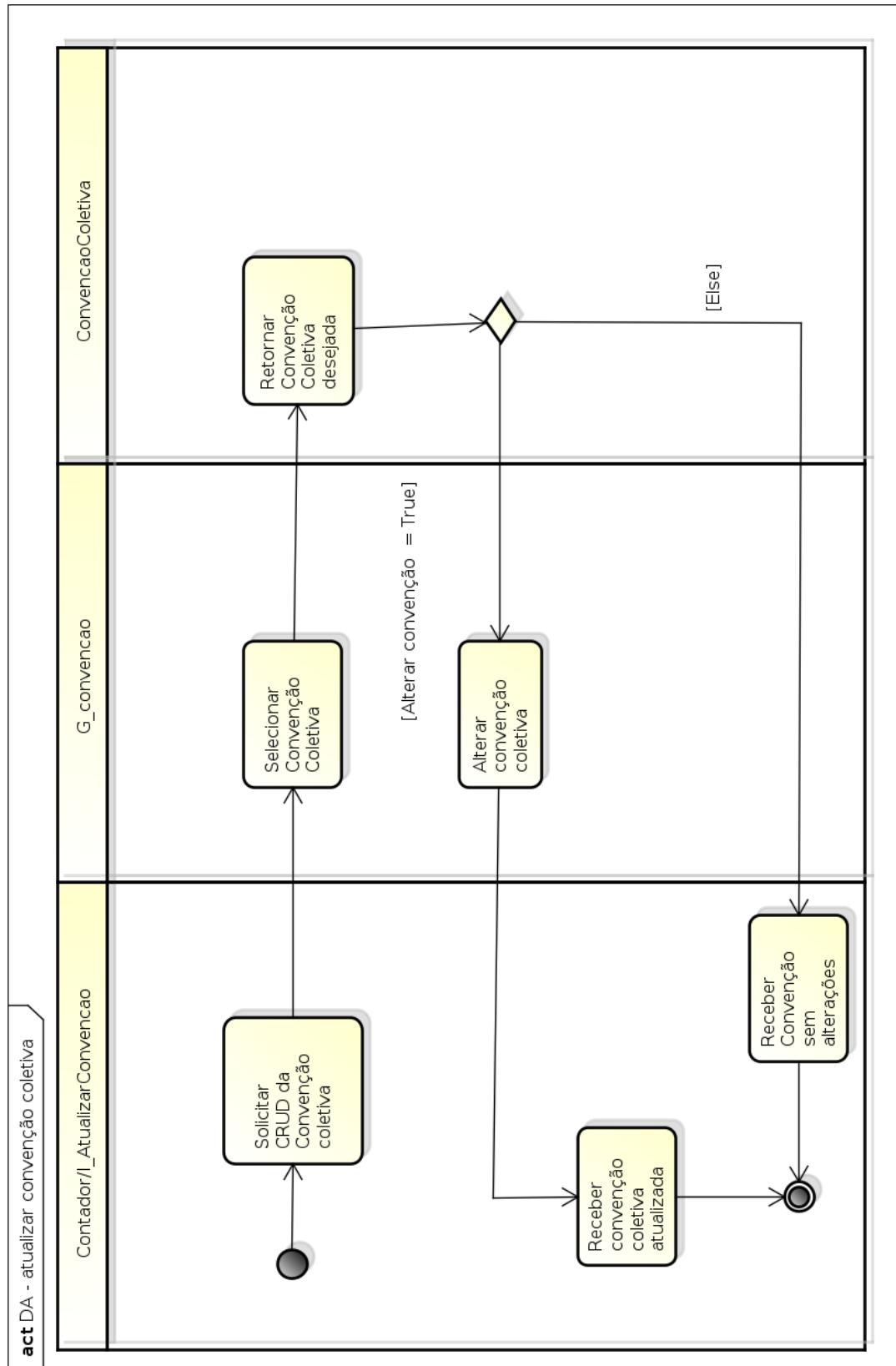


Figura 44 – Diagrama de Atividades de Atualizar convenção coletiva (Kevin)

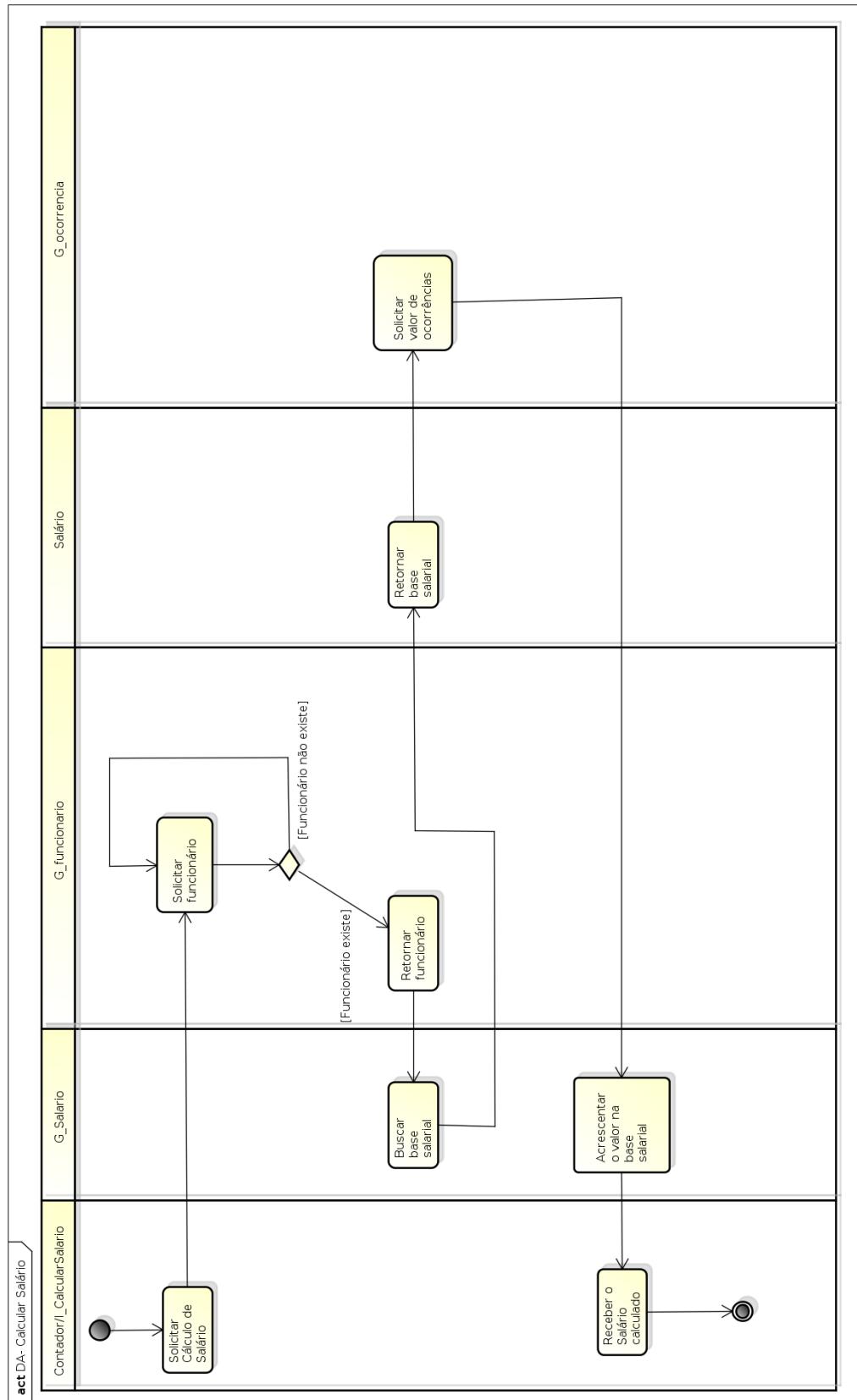


Figura 45 – Diagrama de Atividades de Calcular salário (Kevin)

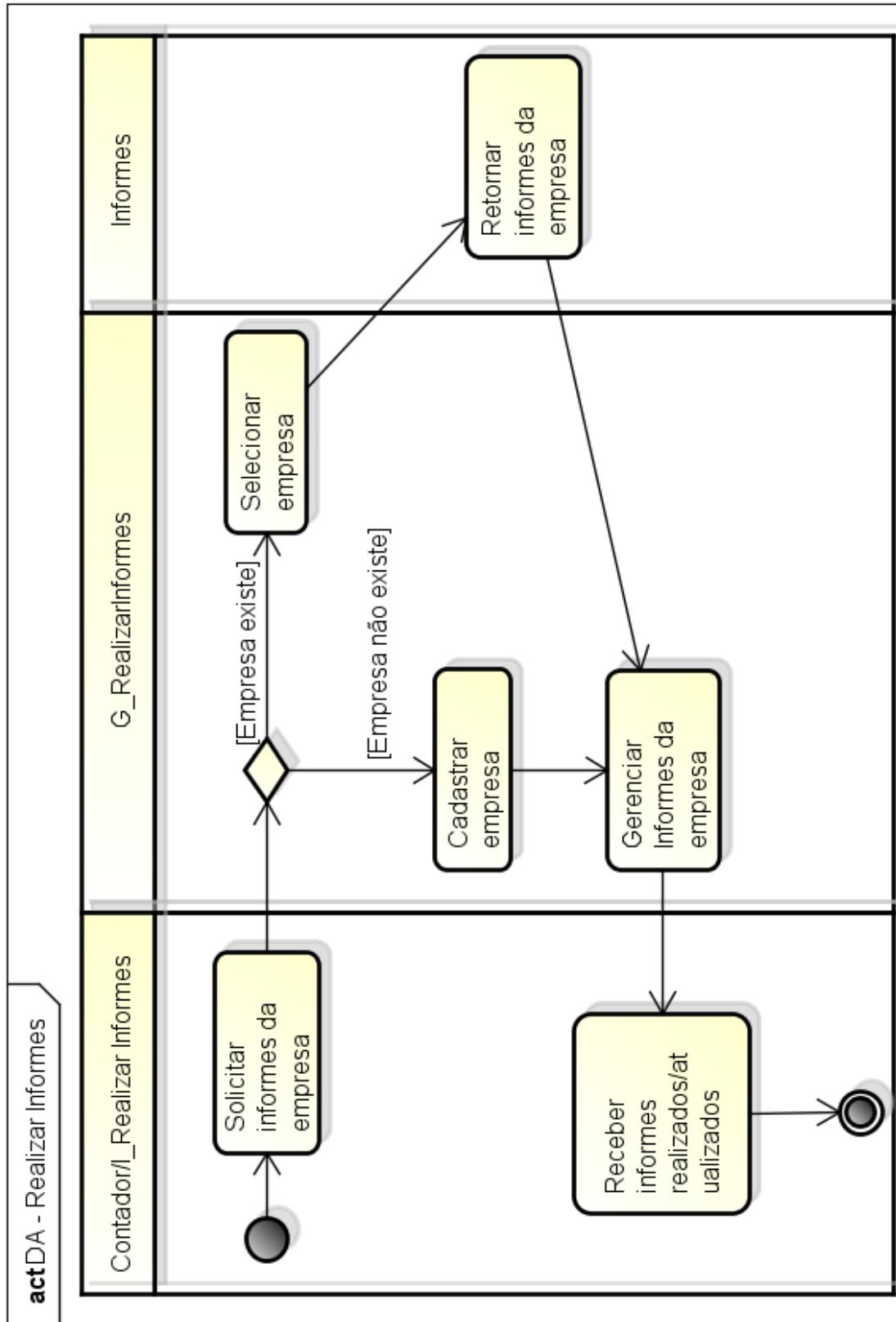


Figura 46 – Diagrama de Atividades de Realizar Informes (Christian)

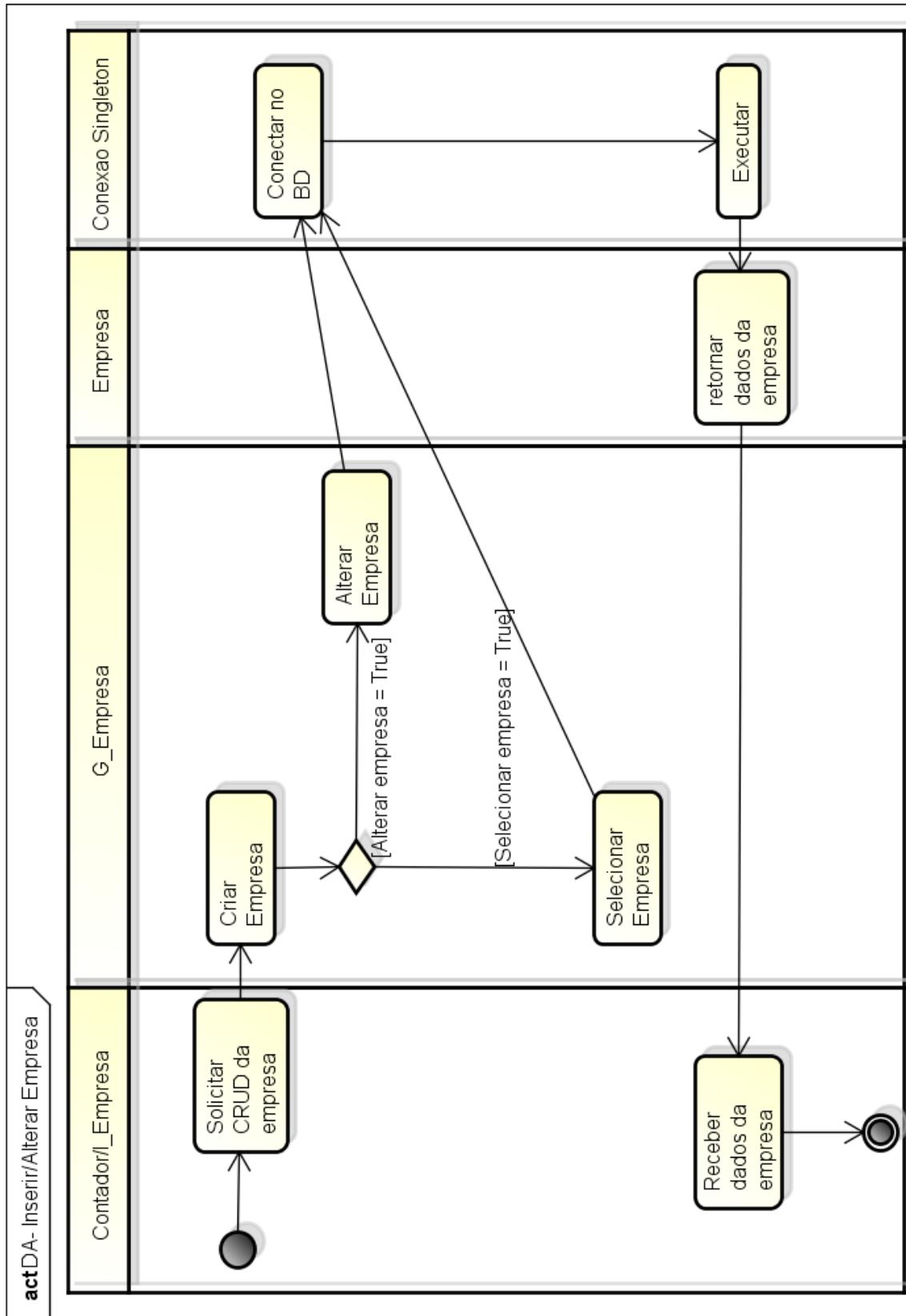


Figura 47 – Diagrama de Atividades de Inserir/Alterar empresa (Christian)

14 Planejamento das SPRINTS

1ª SPRINT

Item	Descrição	Tempo estimado	Quem
Muito Importante			
4	Terminar conexão com banco de dados	48 horas	Pedro
Importante			
*	CRUDs		
2	Manter Contrato	48 horas	Alessandra
3	Manter Funcionário	48 horas	Alessandra
4	Manter Convenção Coletiva dos Sindicatos	48 horas	Christian
5	Manter Sindicatos	48 horas	Christian
6	Manter Ocorrências	48 horas	Kevin
7	Manter Aviso Prédio	48 horas	Kevin
8	Manter Empresa	48 horas	Pedro

2ª SPRINT

Item	Descrição	Tempo estimado	Quem
Importante			
1	Revisar banco de dados	120 hrs	Todos
2	Revisar tarefas da SPRINT anterior	120 hrs	Todos
3	Implementação do Padrão Singleton	120 hrs	Todos
4	Implementação do Observer	120 hrs	Todos
5	Calcular Salário	168 hrs	Kevin
6	Realizar Informes de Imposto de Renda	168 hrs	Alessandra
7	Emitir Holerite (Folha de Pagto)	168 hrs	Kevin
8	Calcular Férias	168 hrs	Christian
9	Calcular Abono Pecuniário	168 hrs	Christian
10	Gerar Relatórios	168 hrs	Alessandra

Figura 48 – Tabela de planejamento das SPRINTS

15 Diagrama de Classes MVC

As classes destacadas escuras (em rosa) são as classes não implementadas. Foram implementados apenas alguns métodos em G_Empresa que foram necessários para implementação de outras classes. Nas figuras 49 e 50 estão simplificadas em classes de análise com objetivo de representar as ligações.

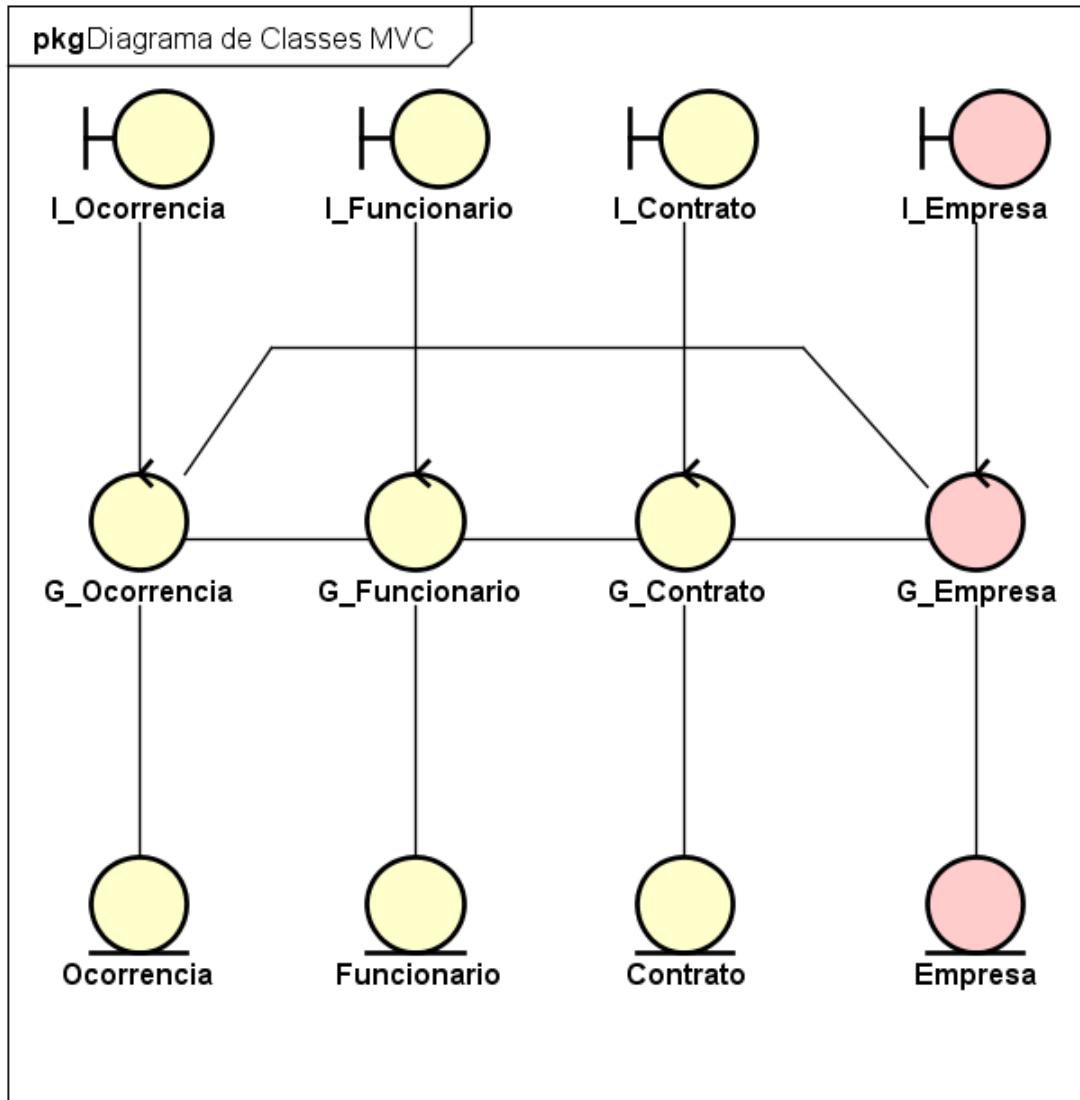


Figura 49 – Diagrama de Classes MVC ligações parte 1

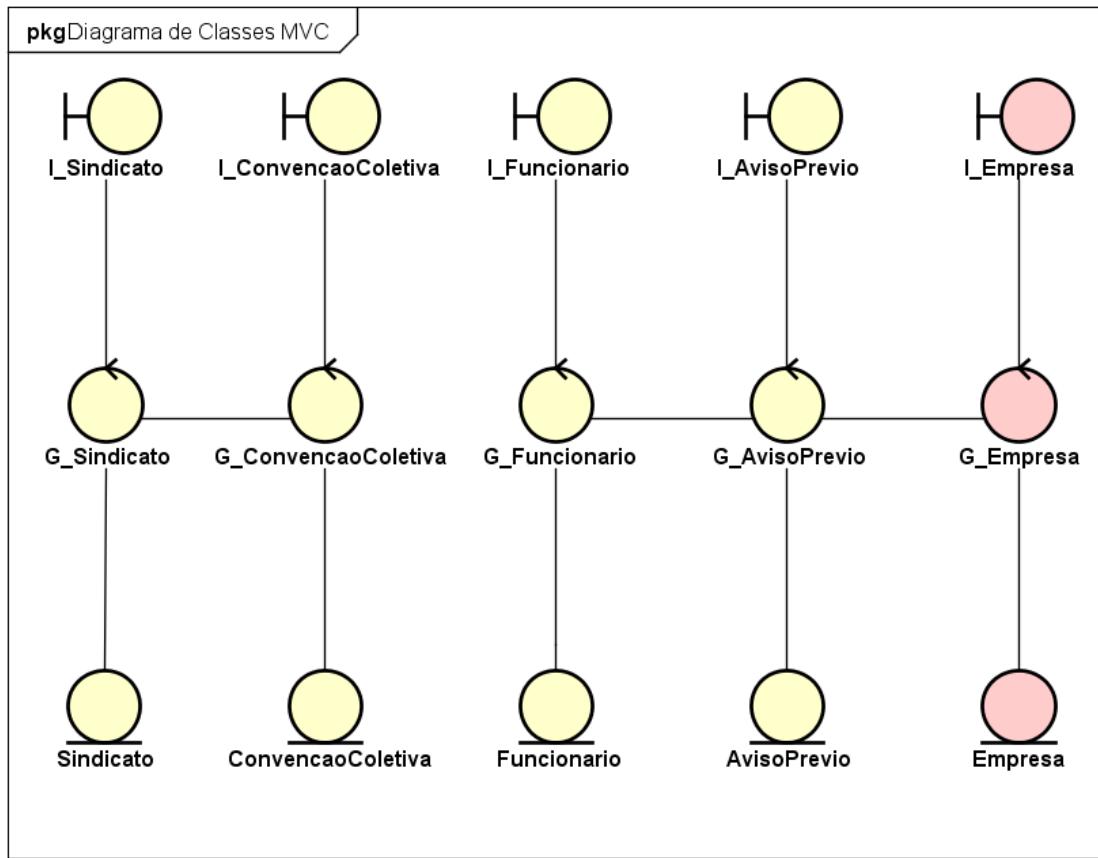


Figura 50 – Diagrama de Classes MVC ligações parte 2

Na imagem ao lado, é possível observar que o Observer fica observando a classe RegistroAtividades. Toda vez que ocorre uma alteração, será verificado se as horas de entrada e saída batem com as constadas no contrato. Caso haja alguma divergência, o Observer cria uma ocorrência para aquele funcionário.

A classe principal Ribanceira é quem chama o Singleton para criar a instância única da conexão com o banco de dados, antes de chamar as janelas com as quais o usuário irá interagir.

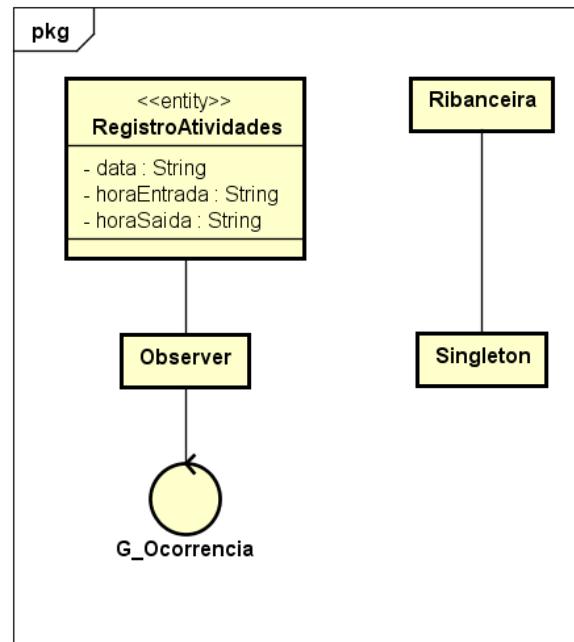


Figura 51 – Diagrama de Classes MVC ligações parte 3

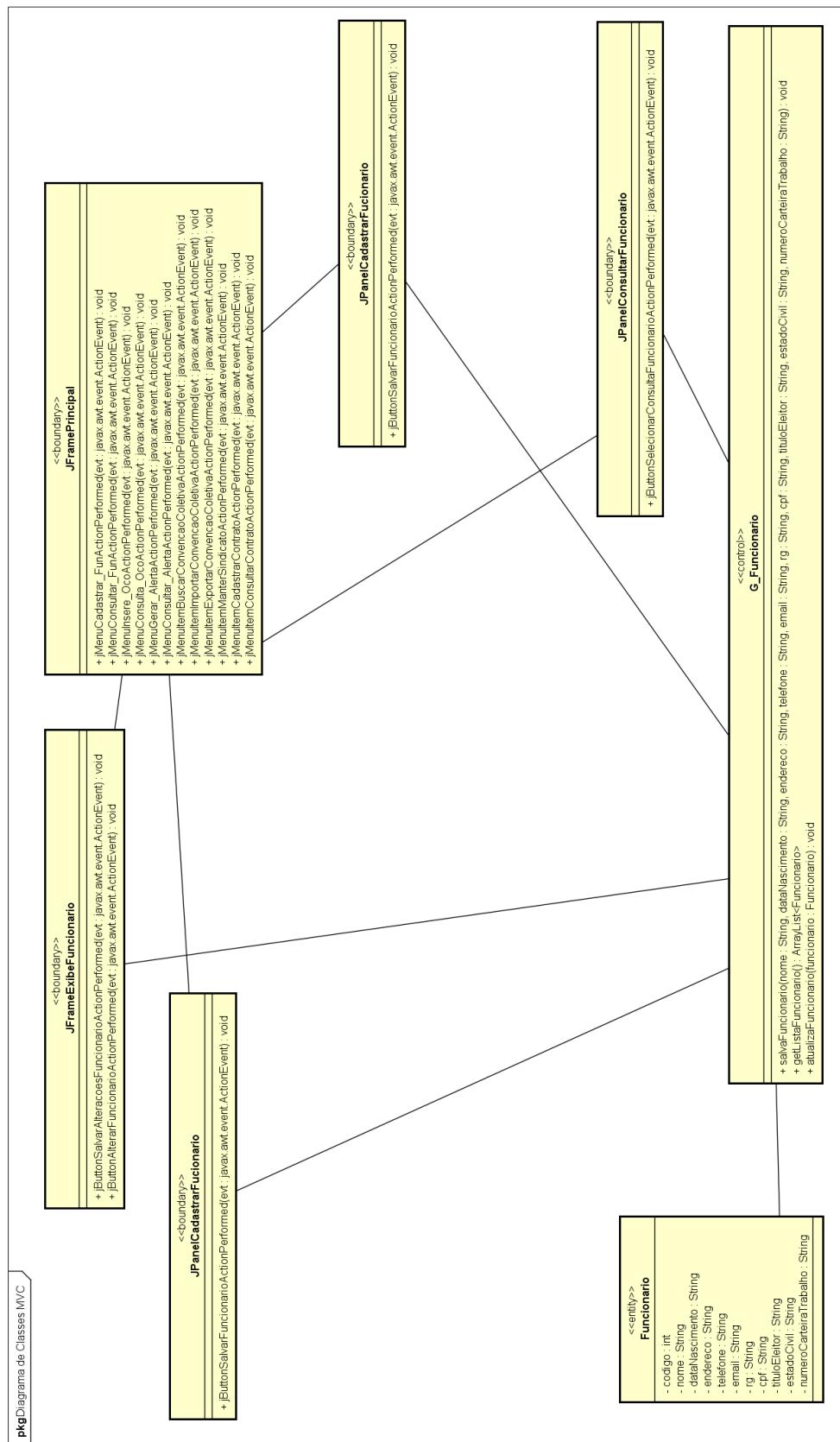


Figura 52 – Diagrama de Classes MVC Funcionário

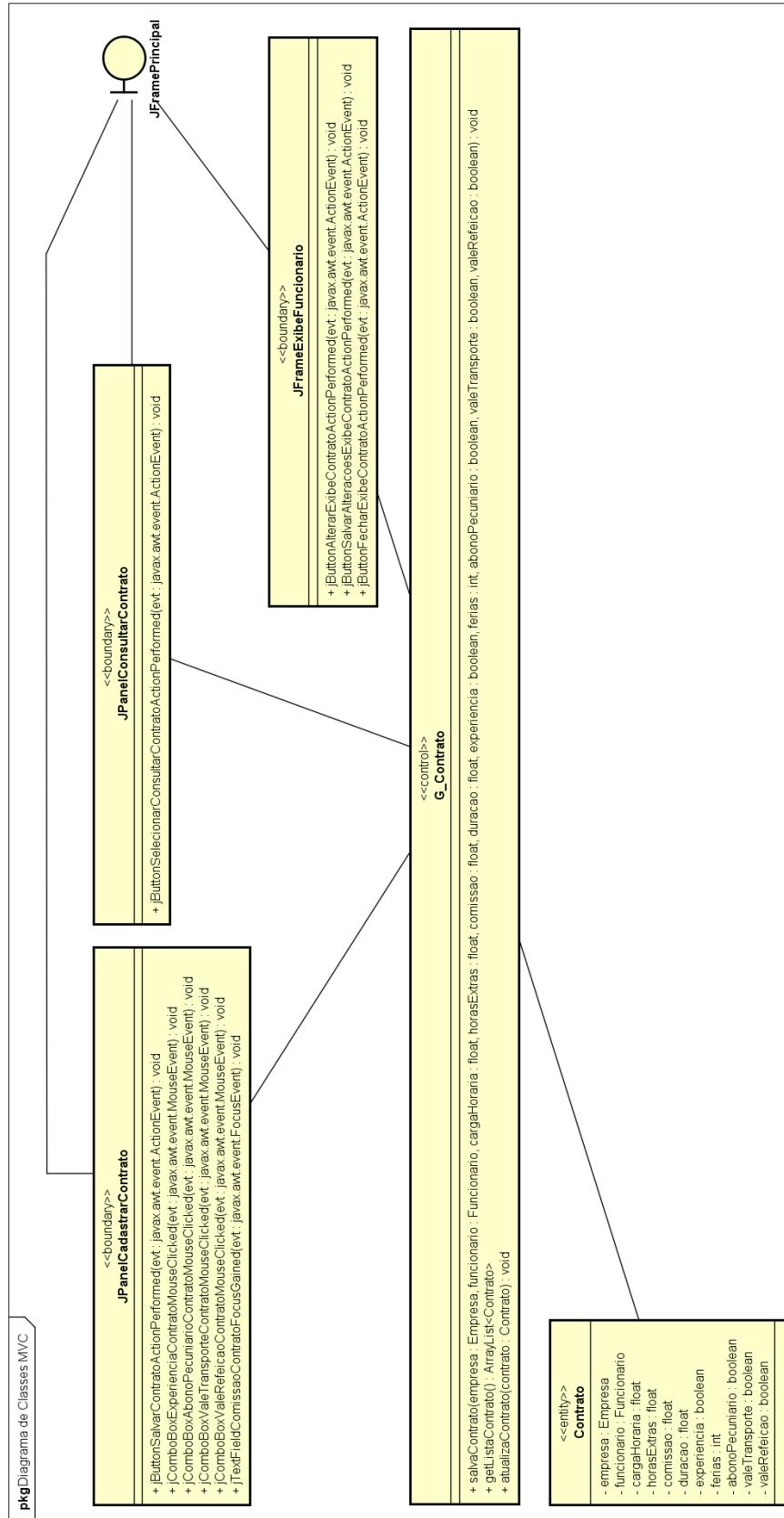


Figura 53 – Diagrama de Classes MVC Contrato

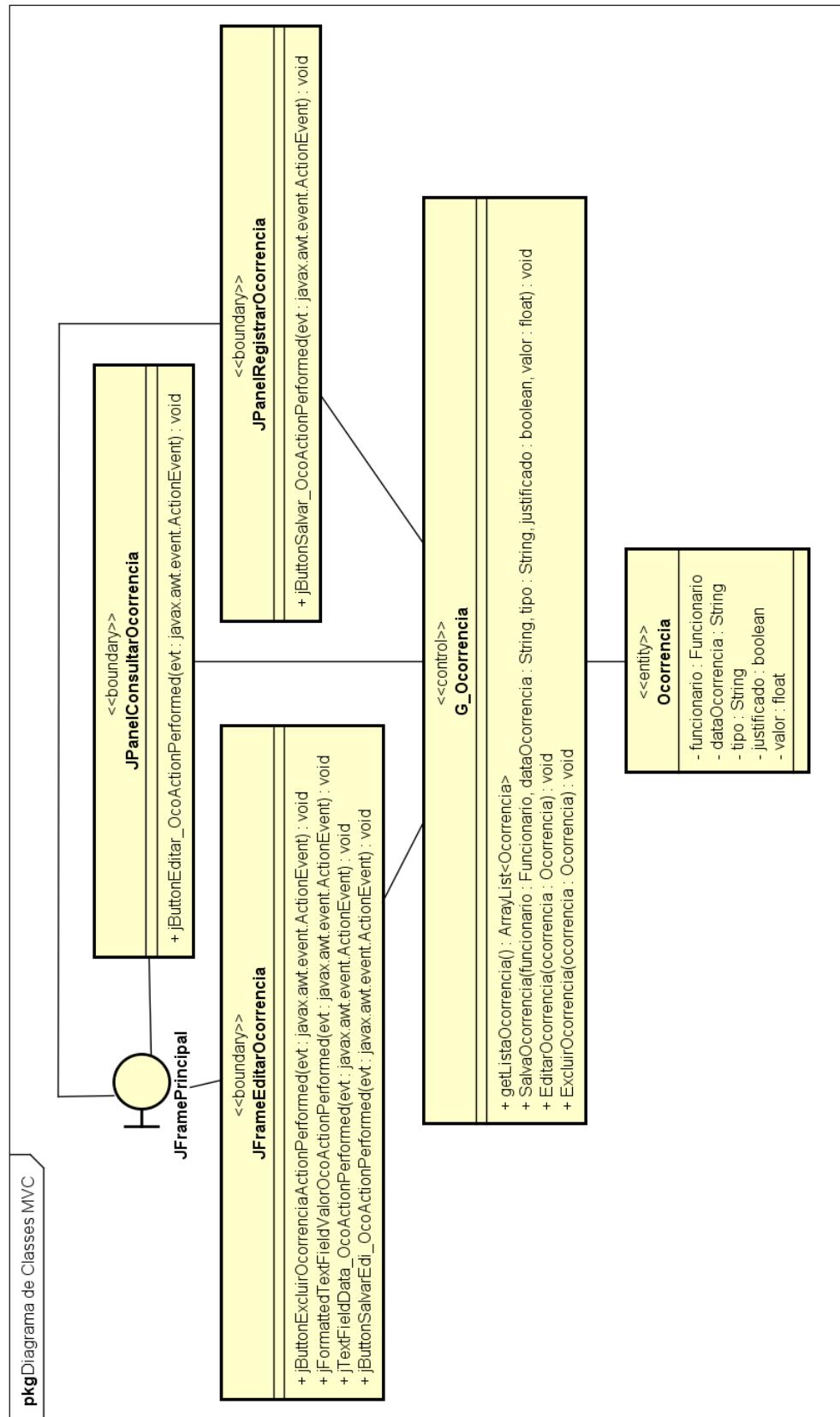


Figura 54 – Diagrama de Classes MVC Ocorrência

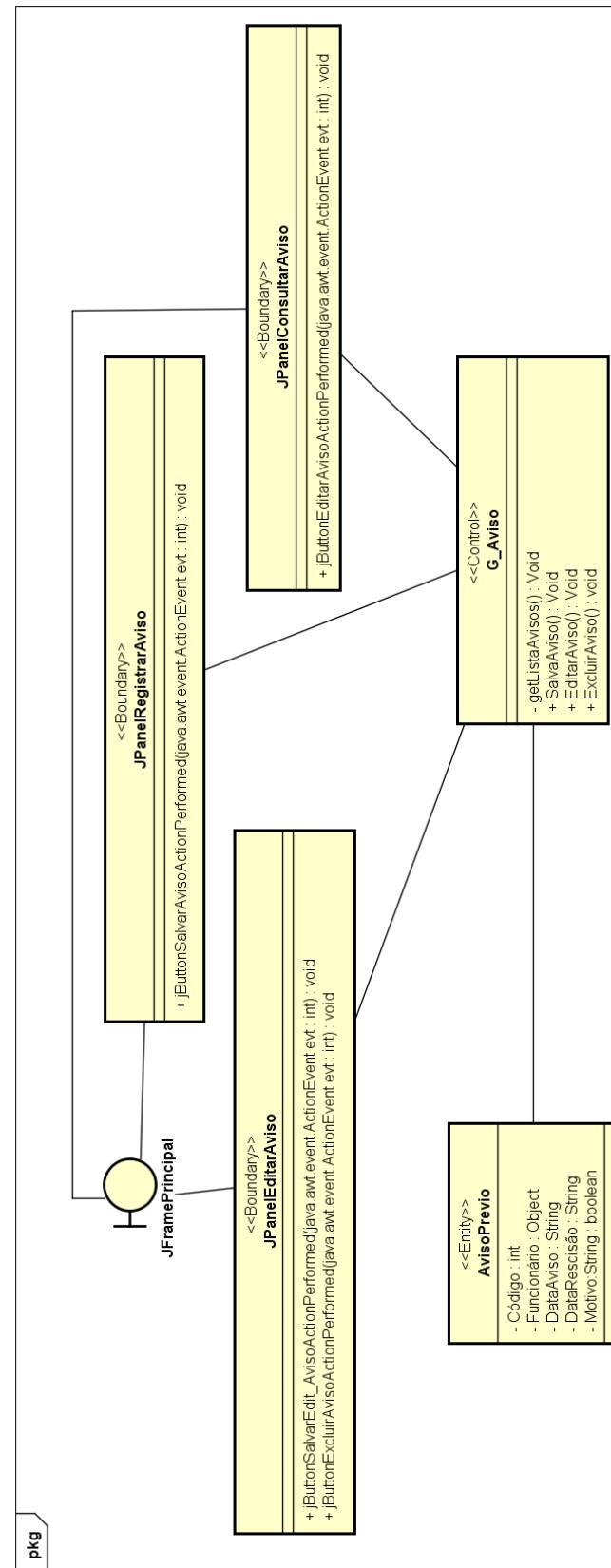


Figura 55 – Diagrama de Classes MVC Aviso Prévio

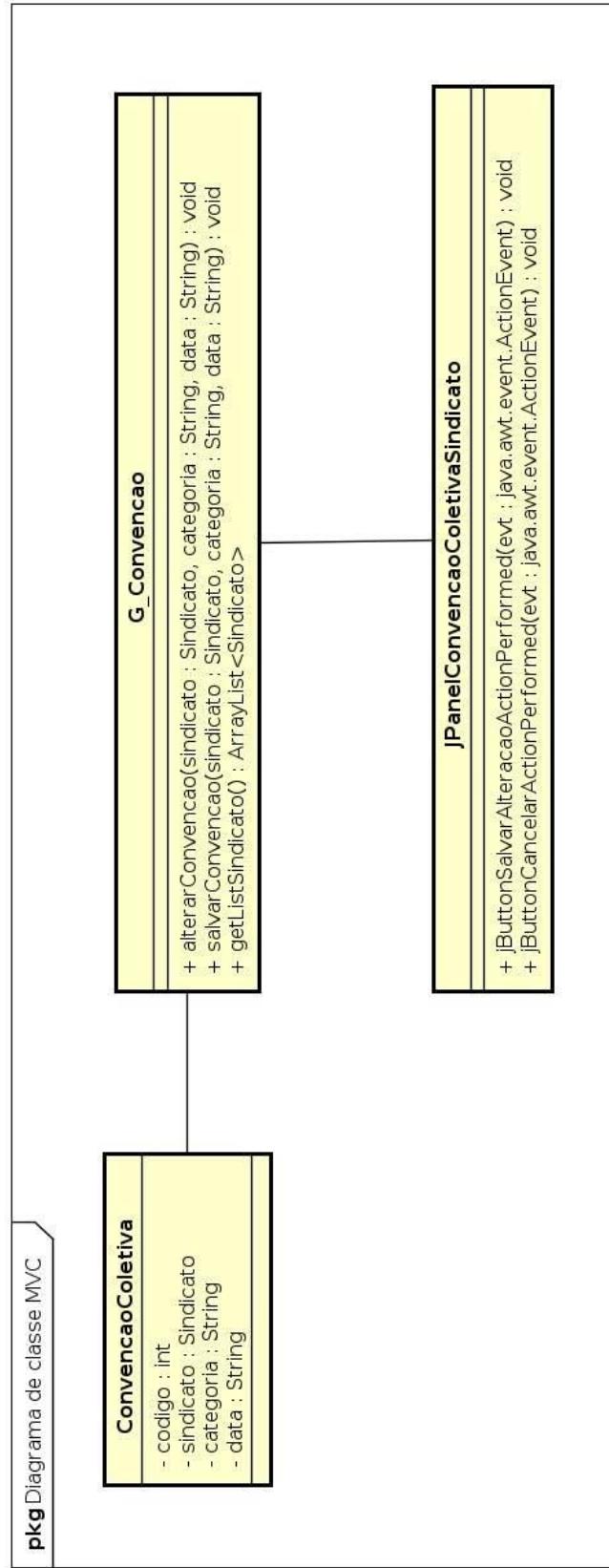


Figura 56 – Diagrama de Classes MVC Convenção Coletiva

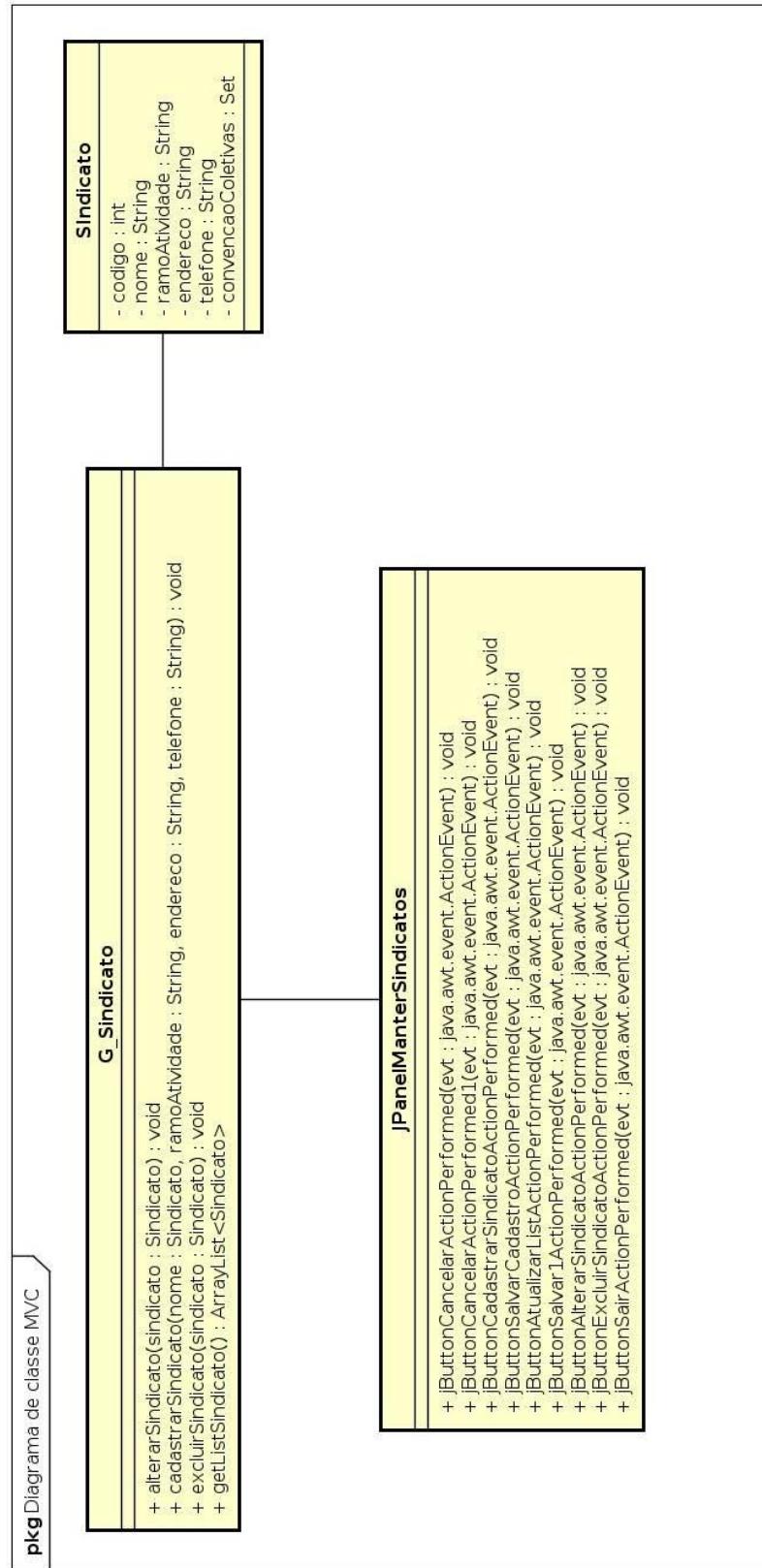


Figura 57 – Diagrama de Classes MVC Sindicato

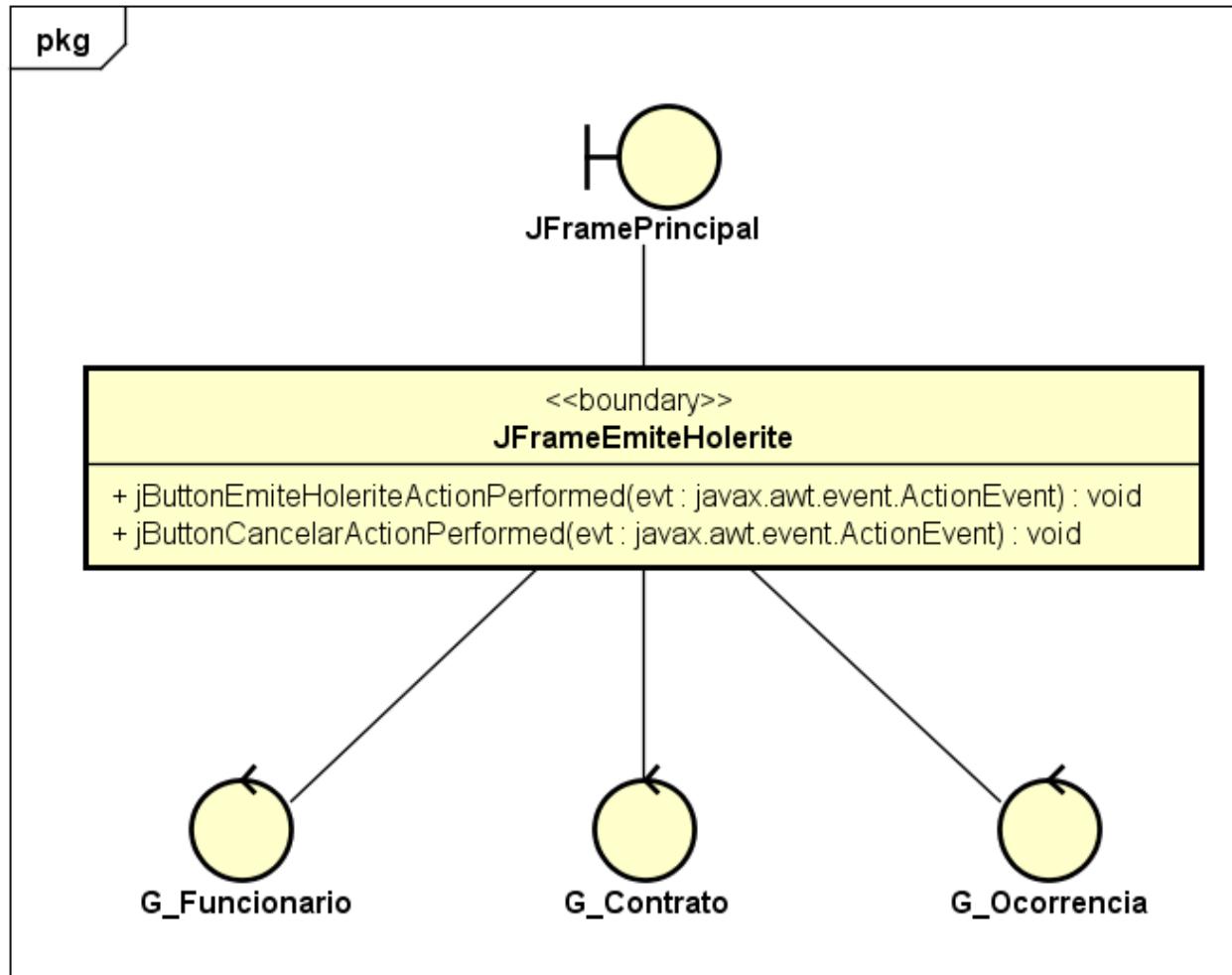


Figura 58 – Diagrama de Classes MVC Sindicato

16 Framework

O *framework* a ser utilizado será o *Hibernate*.

Devido à má documentação do *framewok MySQL++* e pouca afinidade com a linguagem C++, foi decidido alterar a linguagem e, consequentemente, o *framework* e a ferramenta para a construção de interfaces. Agora, trabalharemos com a linguagem Java no ambiente de desenvolvimento integrado NetBeans (versões 8.1 e 8.2). Para a conexão com o banco de dados e alterações no banco de dados, será utilizado o *framework Hibernate*. Por fim, para a construção das interfaces, será utilizado o *widget toolkit GUI* conhecido como *swing*.

Na figura 57, observa-se que as telas criadas, e nomeadas de forma genérica, são alguns dos componentes de *swing* do Java. Observa-se, também, a representação do *Hibernate* que realiza a conexão com o banco de dados. O *Singleton* serve para criar apenas uma instância dessa conexão. O *Observer* serve para realizar o cadastro de Ocorrências a partir de uma alteração no Registro de Atividades.

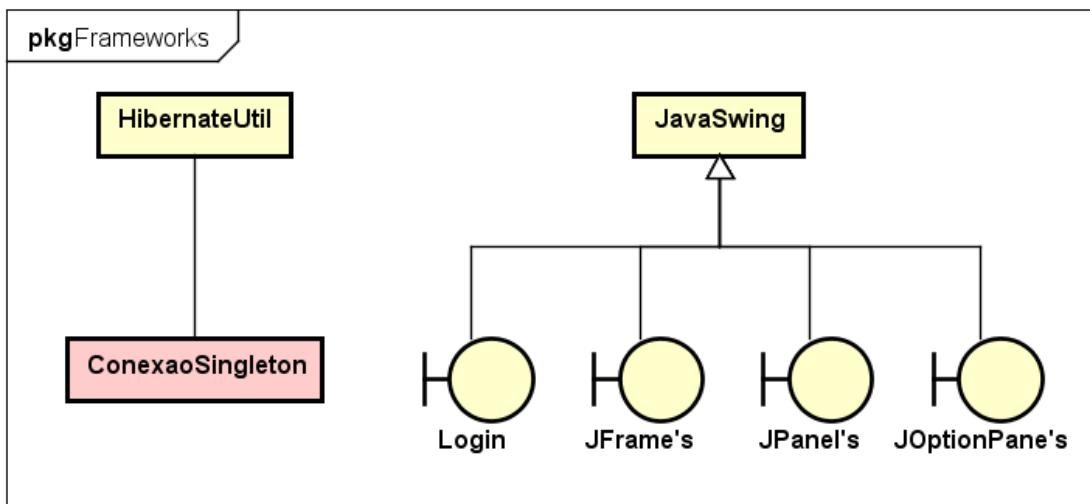


Figura 59 – Diagrama de Classes *Frameworks*

17 Padrões de Projeto

Um padrão de projeto a ser utilizado, será o *Singleton* e um outro será o *Observer*. O *Singleton* será utilizado para criar uma instância da conexão com o banco de dados e esta será única. Enquanto o *Observer* será utilizado para que, toda vez que acontecer uma alteração na classe de Registro de Atividades, seja criado uma ocorrência para o funcionário de forma apropriada. Estes podem ser observados na Figura 51.

O *Singleton* não foi utilizado para realização de cadastros, consultas, etc. Ele foi utilizado para apenas criar uma única instância da conexão com o banco de dados. Isso se deve ao fato dos integrantes terem declarado o uso do Hibernate mais fácil. Além de, por terem contato há mais tempo, terem mais afinidade. Caso a conexão não seja instanciada com sucesso pelo *Singleton*, a aplicação se encerra.

Já o padrão do *Observer* foi implementado, porém houve alguns problemas de uso e não foi possível completar o uso dele.

18 Testes

18.1. Caixa branca

18.1.1. Alessandra

Método `jButtonGerarRelatorioActionPerformed` (`java.awt.event.ActionEvent evt`), presente em `JFrameRelatoriosGerais.java`. Segue o código deste método:

```

1  private void jButtonGerarRelatorioActionPerformed(java.awt.event.ActionEvent evt) {
2      // TODO add your handling code here:
3      if(jComboBoxCriterioRelatorio.getSelectedIndex() == 0) {
4          Document doc = new Document();
5          Empresa empresa = listaEmpresa.get(jComboBoxNomeRelatorio.getSelectedIndex());
6          Set<Contrato> listaContratoEmpresa = empresa.getContratos();
7          String nomeArquivo = empresa.getNomeEmpresa() + new Date().getTime() + ".pdf";
8
9          if(jRadioButtonAtivoRelatorio.isSelected()) {
10             try {
11                 PdfWriter.getInstance(doc, new FileOutputStream(nomeArquivo));
12                 doc.open();
13                 doc.add(new Paragraph("RELATÓRIO: Lista de funcionário de " + empresa.getNomeEmpresa()));
14                 doc.add(new Paragraph("CNPJ: " + empresa.getCnpj()));
15                 doc.add(new Paragraph("-----"));
16                 for(Contrato c : listaContratoEmpresa) {
17                     if(c.getFuncionario().isAtivo()) {
18                         doc.add(new Paragraph("Nome: " + c.getFuncionario().getNome()));
19                         if(jCheckBoxDado2.isSelected()) {
20                             doc.add(new Paragraph("Data de nascimento: " + c.getFuncionario().getDataNascimento()));
21                         }
22                         if(jCheckBoxDado3.isSelected()) {
23                             doc.add(new Paragraph("Endereço: " + c.getFuncionario().getEndereco()));
24                         }
25                         if(jCheckBoxDado4.isSelected()) {
26                             doc.add(new Paragraph("Telefone: " + c.getFuncionario().getTelefone()));
27                         }
28                         if(jCheckBoxDado5.isSelected()) {
29                             doc.add(new Paragraph("E-mail: " + c.getFuncionario().getEmail()));
30                         }
31                         if(jCheckBoxDado6.isSelected()) {
32                             doc.add(new Paragraph("RG: " + c.getFuncionario().getRg()));
33                         }
34                         if(jCheckBoxDado7.isSelected()) {
35                             doc.add(new Paragraph("CPF: " + c.getFuncionario().getCpf()));
36                         }
37                         if(jCheckBoxDado8.isSelected()) {
38                             doc.add(new Paragraph("Carteira de trabalho: " + c.getFuncionario().getNumeroCarteiraTrabalho()));
39                         }
40                     doc.add(new Paragraph("-----"));
41                 }
42             } catch (DocumentException | FileNotFoundException ex) {
43                 System.out.println("Error: " + ex);
44             } finally {
45                 doc.close();
46             }
47         try {
48             Desktop.getDesktop().open(new File(nomeArquivo));
49         } catch (IOException ex) {
50             System.out.println("Error: " + ex);
51         }
52     }
53     else { // funcionários não ativos
54         try {
55             PdfWriter.getInstance(doc, new FileOutputStream(nomeArquivo));
56             doc.open();
57             doc.add(new Paragraph("RELATÓRIO: Lista de funcionário inativos de " + empresa.getNomeEmpresa()));
58             doc.add(new Paragraph("CNPJ: " + empresa.getCnpj()));
59             doc.add(new Paragraph("-----"));
60             for(Contrato c : listaContratoEmpresa) {
61                 if(!c.getFuncionario().isAtivo()) {
62                     doc.add(new Paragraph("Nome: " + c.getFuncionario().getNome()));
63                     if(jCheckBoxDado2.isSelected()) {
64                         doc.add(new Paragraph("Data de nascimento: " + c.getFuncionario().getDataNascimento()));
65                     }
66                     if(jCheckBoxDado3.isSelected()) {
67                         doc.add(new Paragraph("Endereço: " + c.getFuncionario().getEndereco()));
68                     }
69                     if(jCheckBoxDado4.isSelected()) {
70                         doc.add(new Paragraph("Telefone: " + c.getFuncionario().getTelefone()));
71                     }
72                     if(jCheckBoxDado5.isSelected()) {
73                         doc.add(new Paragraph("E-mail: " + c.getFuncionario().getEmail()));
74                     }
75                     if(jCheckBoxDado6.isSelected()) {
76                         doc.add(new Paragraph("RG: " + c.getFuncionario().getRg()));
77                     }
78                     if(jCheckBoxDado7.isSelected()) {
79                         doc.add(new Paragraph("CPF: " + c.getFuncionario().getCpf()));
80                     }
81                 }
82             }
83         }
84     }
85 }

```

```
81         }
82         if(jCheckBoxDado8.isSelected()) {
83             doc.add(new Paragraph("Carteira de trabalho: " + c.getFuncionario().getNumeroCarteiraTrabalho()));
84         }
85         doc.add(new Paragraph("-----"));
86     }
87 }
88 } catch (DocumentException | FileNotFoundException ex) {
89     System.out.println("Error: " + ex);
90 } finally {
91     doc.close();
92 }
93 try {
94     Desktop.getDesktop().open(new File(nomeArquivo));
95 } catch (IOException ex) {
96     System.out.println("Error: " + ex);
97 }
98 }
99 }
100 else { // imprimir relatório de dados do funcionário
101     Contrato contrato = listaContratos.get(jComboBoxNomeRelatorio.getSelectedIndex());
102     String nomeArquivo = contrato.getFuncionario().getNome() + new Date().getTime() + ".pdf";
103     Document doc = new Document();
104
105     if(jComboBoxCriterioFuncionarioRelatorio.getSelectedIndex() == 0) {
106         try {
107             PdfWriter.getInstance(doc, new FileOutputStream(nomeArquivo));
108             doc.open();
109             doc.add(new Paragraph("RELATÓRIO: Dados do contrato de " + contrato.getFuncionario().getNome()
110                                 + " inscrito sob o CPF: " + contrato.getFuncionario().getCpf()));
111             doc.add(new Paragraph("com a empresa: " + contrato.getEmpresa().getNomeEmpresa()));
112             doc.add(new Paragraph(" "));
113             if(jCheckBoxDado1.isSelected()) {
114                 doc.add(new Paragraph("Carga horária: " + Float.toString(contrato.getCargaHoraria())));
115             }
116             if(jCheckBoxDado2.isSelected()) {
117                 doc.add(new Paragraph("Hora entrada: " + contrato.getHoraEntrada()));
118             }
119             if(jCheckBoxDado3.isSelected()) {
120                 doc.add(new Paragraph("Hora saída: " + contrato.getHoraSaida()));
121             }
122             if(jCheckBoxDado4.isSelected()) {
123                 doc.add(new Paragraph("Horas extras: " + Float.toString(contrato.getHorasExtras())));
124             }
125             if(jCheckBoxDado5.isSelected()) {
126                 doc.add(new Paragraph("Comissão: " + Float.toString(contrato.getComissao())));
127             }
128             if(jCheckBoxDado6.isSelected()) {
129                 doc.add(new Paragraph("Duração: " + Float.toString(contrato.getDuracao())));
130             }
131             if(jCheckBoxDado7.isSelected()) {
132                 String exp;
133                 if(contrato.isExperiencia()) {
134                     exp = "Sim";
135                 }
136                 else {
137                     exp = "Não";
138                 }
139                 doc.add(new Paragraph("Experiência: " + exp));
140             }
141             if(jCheckBoxDado8.isSelected()) {
142                 doc.add(new Paragraph("Férias: " + Integer.toString(contrato.getFerias())));
143             }
144             if(jCheckBoxDado9.isSelected()) {
145                 String vt;
146                 if(contrato.isValeTransporte()) {
147                     vt = "Sim";
148                 }
149                 else {
150                     vt = "Não";
151                 }
152                 doc.add(new Paragraph("Vale transporte: " + vt));
153             }
154             if(jCheckBoxDado10.isSelected()) {
155                 String vr;
156                 if(contrato.isValeRefeicao()) {
157                     vr = "Sim";
158                 }
159                 else {
160                     vr = "Não";
161                 }
162                 doc.add(new Paragraph("Vale refeição: " + vr));
163             }
164             if(jCheckBoxDado11.isSelected()) {
165                 doc.add(new Paragraph("Salário base: " + Float.toString(contrato.getBaseSalarial())));
166             }
167             if(jCheckBoxDado12.isSelected()) {
168                 doc.add(new Paragraph("Impostos: " + Float.toString(contrato.getImpostos())));
169             }
170 } catch (DocumentException | FileNotFoundException ex) {
171     System.out.println("Error: " + ex);
172 } finally {
```

```

172     doc.close();
173 }
174 try {
175     Desktop.getDesktop().open(new File(nomeArquivo));
176 } catch (IOException ex) {
177     System.out.println("Error: " + ex);
178 }
179 } else {
180     Set<Ocorrencia> listaOcorrencia = contrato.getFuncionario().getOcorrencias();
181     try {
182         PdfWriter.getInstance(doc, new FileOutputStream(nomeArquivo));
183         doc.open();
184         doc.add(new Paragraph("RELATÓRIO: Ocorrências de " + contrato.getFuncionario().getNome() +
185                             " inscrito sob o CPF: " + contrato.getFuncionario().getCpf()));
186         doc.add(new Paragraph("com a empresa: " + contrato.getEmpresa().getNomeEmpresa()));
187         doc.add(new Paragraph(" "));
188         for(Ocorrencia ocorrencia : listaOcorrencia) {
189             String just;
190             if(ocorrencia.isJustificado()) {
191                 just = "Sim";
192             } else {
193                 just = "Não";
194             }
195             doc.add(new Paragraph("Data: " + ocorrencia.getDataOcorrencia()));
196             doc.add(new Paragraph("Tipo: " + ocorrencia.getTipo()));
197             doc.add(new Paragraph("Justificado " + just));
198             doc.add(new Paragraph("Valor: " + Float.toString(ocorrencia.getValor())));
199             doc.add(new Paragraph("-----"));
200         }
201     } catch (DocumentException | FileNotFoundException ex) {
202         System.out.println("Error: " + ex);
203     } finally {
204         doc.close();
205     }
206 }
207 try {
208     Desktop.getDesktop().open(new File(nomeArquivo));
209 } catch (IOException ex) {
210     System.out.println("Error: " + ex);
211 }
212 }
213 }
214 }

```

Figura 60 – Código do jButtonGerarRelatorioActionPerformed

O grafo do método está dividido nas figuras 61 e 62. A complexidade ciclomática é dada pelo número de arestas subtraído de número vértices somado com dois. Com isso, a complexidade ciclomática é $213 - 167 + 2 = 48$. Abaixo, alguns caminhos independentes:

C1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 40 – 41 – 42.

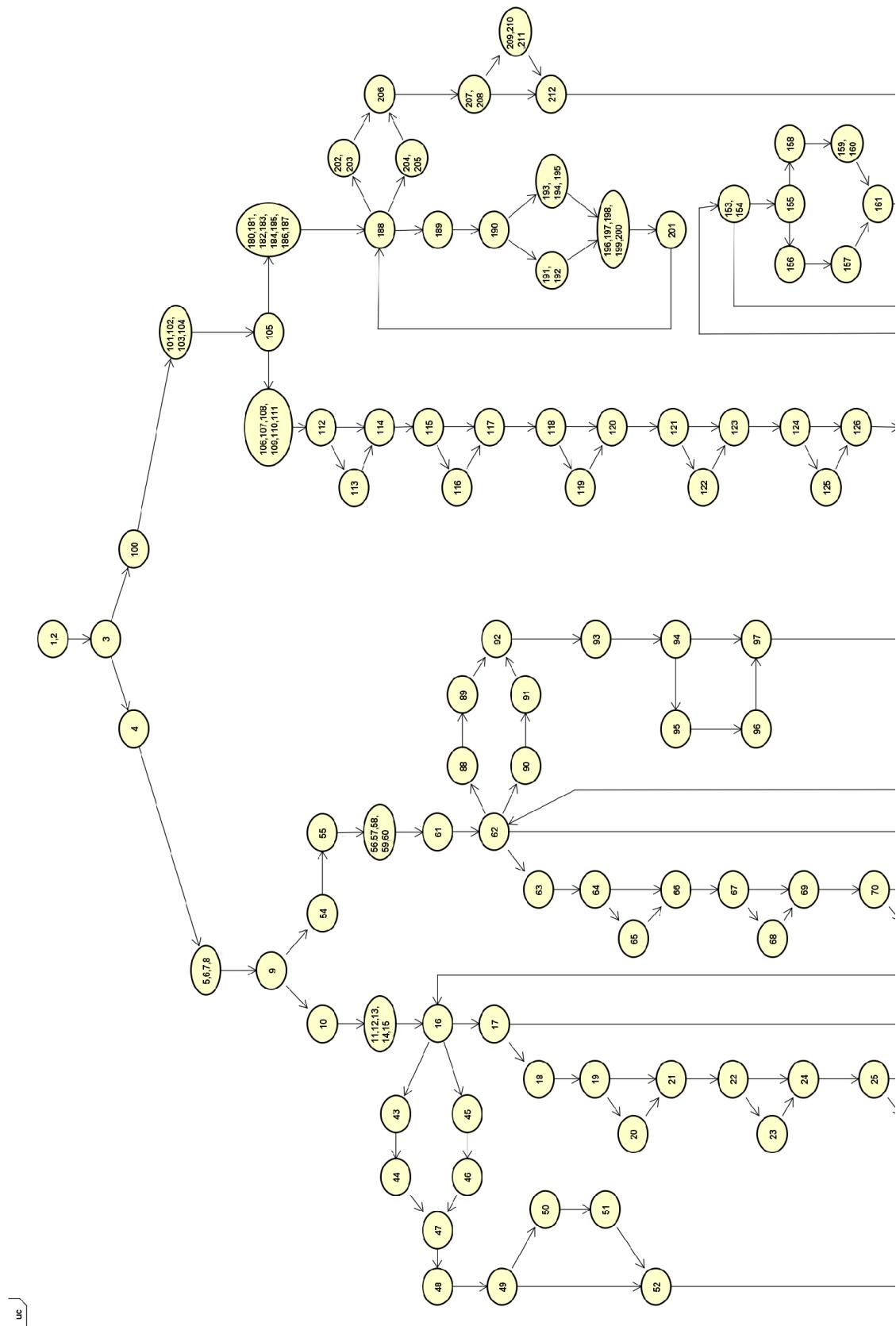
C2: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 43 – 44 – 47 – 48 – 49 – 52 – 53 – 99 – 214.

C3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 45 – 46 – 47 – 48 – 49 – 52 – 53 – 99 – 214.

C4: 1 – 2 – 3 – 100 – 101 – 102 – 103 – 104 – 105 – 106 – 107 – 108 – 109 – 110 – 111 – 112 – 114 – 115 – 116 – 117 – 118 – 119 – 120 – 121 – 123 – 124 – 126 – 127 – 129 – 130 – 131 – 139 – 140 – 142 – 143 – 144 – 152 – 153 – 154 – 155 – 156 – 157 – 161 – 162 – 163 – 165 – 166 – 168 – 171 – 172 – 173 – 174 – 175 – 178 – 179 – 213 – 214.

C5: 1 – 2 – 3 – 100 – 101 – 102 – 103 – 104 – 105 – 180 – 181 – 182 – 183 – 184 – 185 – 186 – 187 – 188 – 189 – 190 – 193 – 194 – 195 – 196 – 197 – 198 – 199 – 200 – 201.

C6: 1 – 2 – 3 – 100 – 101 – 102 – 103 – 104 – 105 – 180 – 181 – 182 – 183 – 184 – 185 – 186 – 187 – 188 – 204 – 205 – 206 – 207 – 208 – 212 – 213 – 214.

Figura 61 – Grafo do método `jButtonGerarRelatorioActionPerformed` parte 1

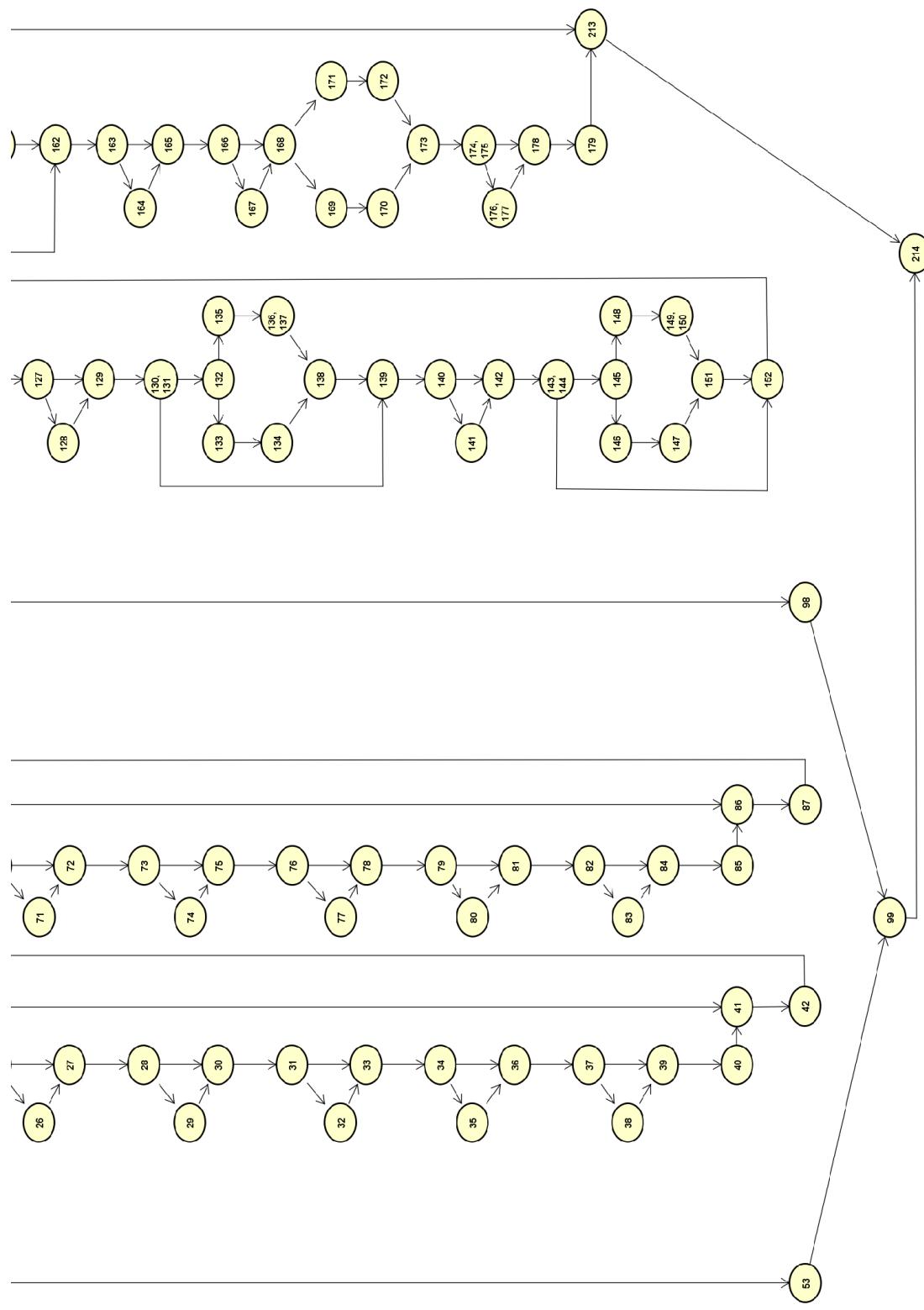


Figura 62 – Grafo do método jButtonGerarRelatorioActionPerformed parte 2

18.1.2. Kevin

Método jButtonEmiteHoleriteActionPerformed (java.awt.event.ActionEvent evt), presente em JFrameEmiteHolerite.java. Segue o código deste método:

```

1  private void jButtonEmiteHoleriteActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jButtonEmiteHoleriteActionPerformed
2      // TODO add your handling code here:
3      Funcionario f;
4      f = listaFuncionario.get(jComboBoxSelecionaFun.getSelectedIndex());
5      Set<Contrato> listaContrato = f.getContratos();
6      JOptionPane.showMessageDialog(null, "Holerite emitido com sucesso!");
7      Document holerite = new Document();
8      try {
9          Font font1 = new Font(Font.FontFamily.TIMES_ROMAN, 8);
10         Font font2 = new Font(Font.FontFamily.TIMES_ROMAN, 14);
11         Font font3 = new Font(Font.FontFamily.TIMES_ROMAN, 20);
12         Font font4 = new Font(Font.FontFamily.TIMES_ROMAN, 11);
13         PdfWriter.getInstance(holerite, new FileOutputStream("Holerite.pdf"));
14         for(Contrato c : listaContrato) {
15             holerite.open();
16             holerite.add(new Paragraph("-----", font2));
17             holerite.add(new Paragraph("Recibo de Pagamento e Salário", font3));
18             holerite.add(new Paragraph(" referente ao mês/ano: " + getDateTime(), font1));
19             holerite.add(new Paragraph("Empregador ", font2));
20             holerite.add(new Paragraph(" Nome: " + c.getEmpresa().getNomeEmpresa(), font4));
21             holerite.add(new Paragraph(" Endereço: " + c.getEmpresa().getNomeEmpresa(), font4));
22             holerite.add(new Paragraph(" CNPJ: " + c.getEmpresa().getCnpj() , font4));
23             holerite.add(new Paragraph("-----", font2));
24             holerite.add(new Paragraph(" "));
25             holerite.add(new Paragraph(" Código NOME DO EMPREGADO", font2));
26             holerite.add(new Paragraph(" " + f.getCodigo() + " " + f.getNome(), font4));
27             holerite.add(new Paragraph(" "));
28             holerite.add(new Paragraph("SALÁRIO BASE: " + c.getBaseSalarial(), font4));
29             holerite.add(new Paragraph("DESCONTO VALE TRANSPORTE: 50,00", font4));
30             holerite.add(new Paragraph("DESCONTO VALE REFEIÇÃO: 50,00", font4));
31             holerite.add(new Paragraph("DESCONTO DAS FALTAS EM DIAS: ", font4));
32             holerite.add(new Paragraph("ACRÉSCIMO DAS HORAS EXTRAS: ", font4));
33             holerite.add(new Paragraph("IMPOSTOS: " + c.getImpostos(), font4));
34             holerite.add(new Paragraph(" "));
35             holerite.add(new Paragraph(" "));
36             holerite.add(new Paragraph("TOTAL DE DESCONTOS: ", font4));
37             holerite.add(new Paragraph("SALÁRIO LÍQUIDO: ", font4));
38             holerite.add(new Paragraph("-----", font2));
39             holerite.add(new Paragraph(" "));
40             holerite.add(new Paragraph("DECLARO TER RECEBIDO A IMPORTÂNCIA LÍQUIDA DISCRIMINADA NESTE RECIBO", font4));
41             holerite.add(new Paragraph(" "));
42             holerite.add(new Paragraph("_____/____ DATA _____ ASSINATURA DO FUNCIONÁRIO", font4));
43             holerite.add(new Paragraph("_____", font4));
44         }
45     } catch (DocumentException | FileNotFoundException ex) {
46         System.out.println("Error:" + ex);
47     } finally{
48         holerite.close();
49     }
50 }
```

Figura 63 – Código jButtonEmiteHoleriteActionPerformed

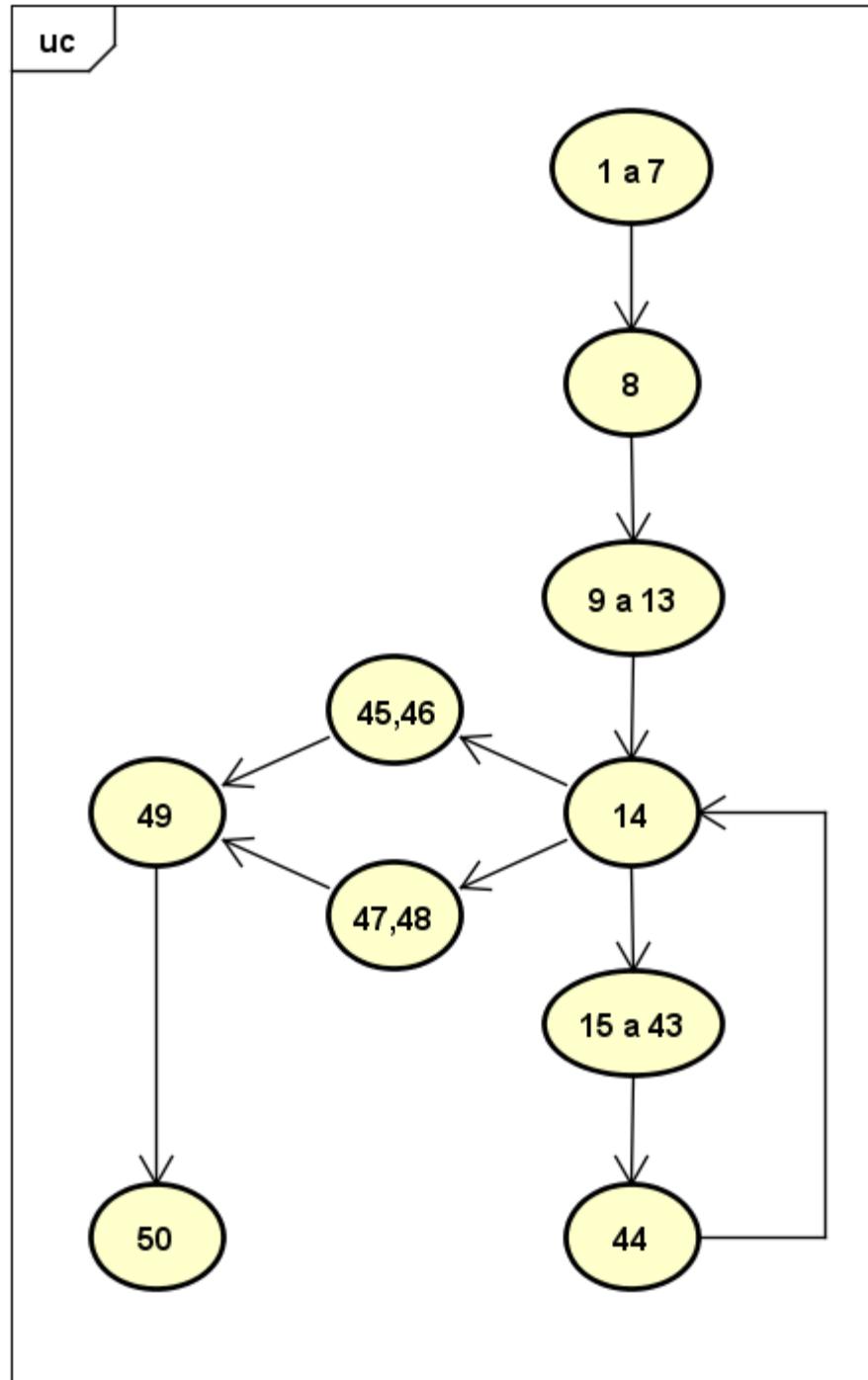
A complexidade ciclomática é dada pelo número de arestas subtraído de número vértices somado com dois. Com isso, a complexidade ciclomática é $11 - 10 + 2 = 3$. Abaixo, alguns caminhos independentes:

C1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 40 – 41 – 42 – 43 – 44.

C2: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 45 – 46 – 49 – 50.

C3: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 47 – 48 – 49 – 50.

O grafo do método se encontra na Figura 64.

Figura 64 – Grafo do método `jButtonEmiteHoleriteActionPerformed`

18.1.3. Christian

Método jButtonSalvarCadastroActionPerformed (java.awt.event.ActionEvent evt), presente em JFrameManterSindicato.java. Segue o código deste método:

```

1  private void jButtonSalvarCadastroActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jButtonSalvarCadastroActionPerformed
2  // TODO add your handling code here:
3  if (!jTextFieldNomeSindicato.getText().equals("") && !jTextFieldRamoAtividade.getText().equals("")) {
4      new G_Sindicato().cadastrarSindicato(jTextFieldCodigoEntidade.getText(),
5          jTextFieldNomeSindicato.getText(),
6          jTextFieldRamoAtividade.getText(),
7          jTextFieldEndereco.getText(),
8          jTextFieldTelefon.getText());
9      JOptionPane.showMessageDialog(this, "Sindicato cadastrado com sucesso!", "Cadastro de Sindicato", JOptionPane.INFORMATION_MESSAGE);
10     this.setVisible(false);
11     jTabbedPaneManterSindicato.setSelectedIndex(0);
12 }
13 else {
14     JOptionPane.showMessageDialog(this, "Error - Falta de dados à inserir", "Erro ao alterar", JOptionPane.INFORMATION_MESSAGE);
15     this.setVisible(false);
16 }
}

```

Figura 65 – Código do método jButtonSalvarCadastroActionPerformed

A complexidade ciclomática é dada pelo número de arestas subtraído de número vértices somado com dois. Com isso, a complexidade ciclomática é igual a $5 - 5 + 2 = 2$. Abaixo, na Figura 66, encontra-se o grafo do método jButtonSalvarCadastroActionPerformed.

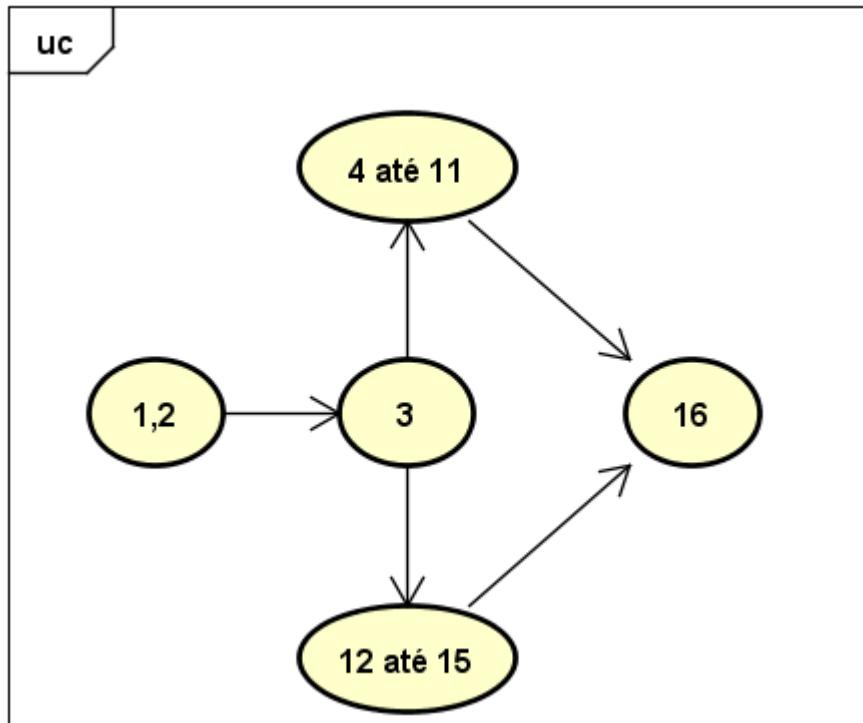


Figura 66 – Grafo do método jButtonSalvarCadastroActionPerformed

Os caminhos independentes são:

C1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 16.

C2: 1 – 2 – 3 – 12 – 13 – 14 – 15 – 16.

19 Telas

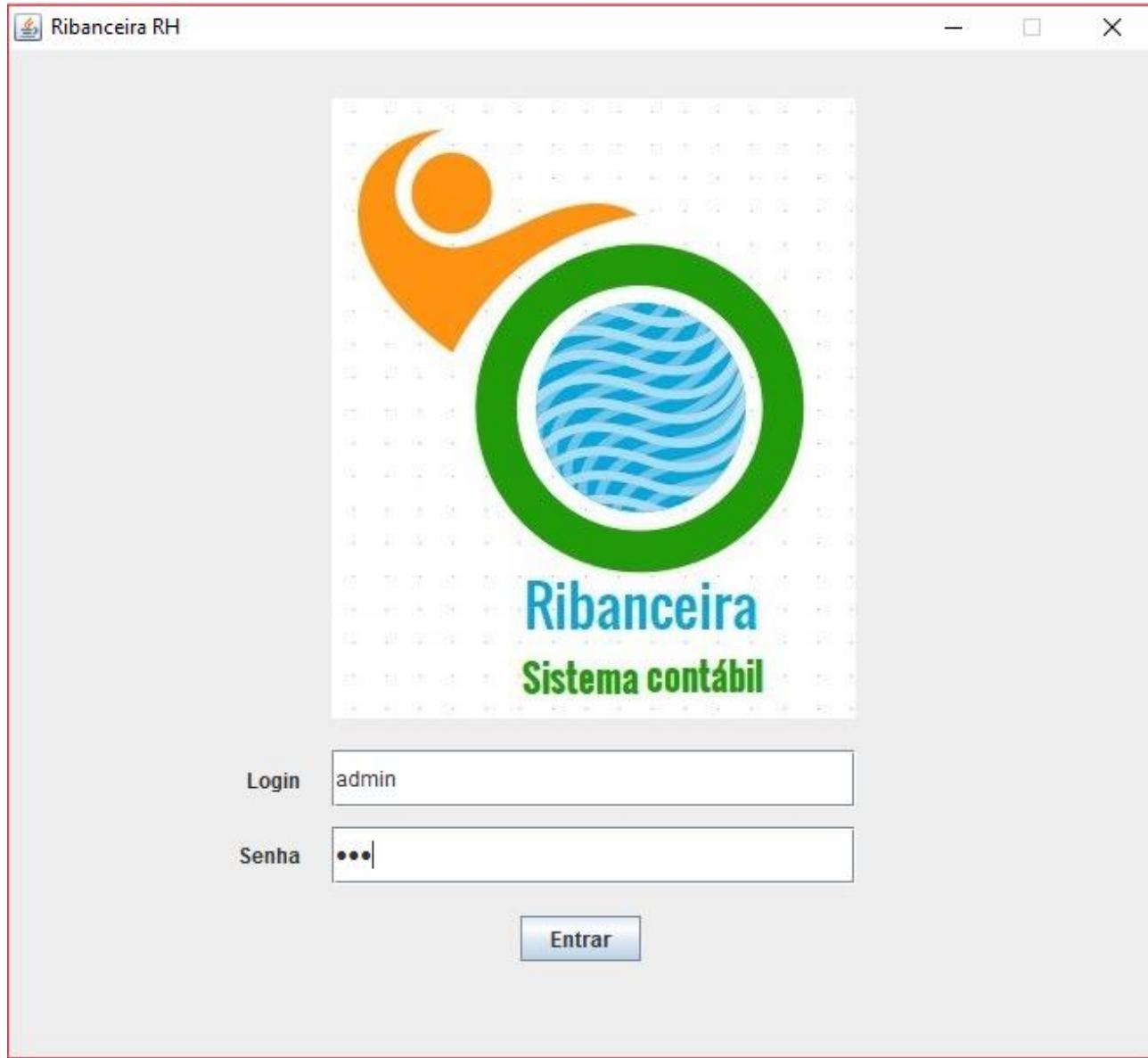


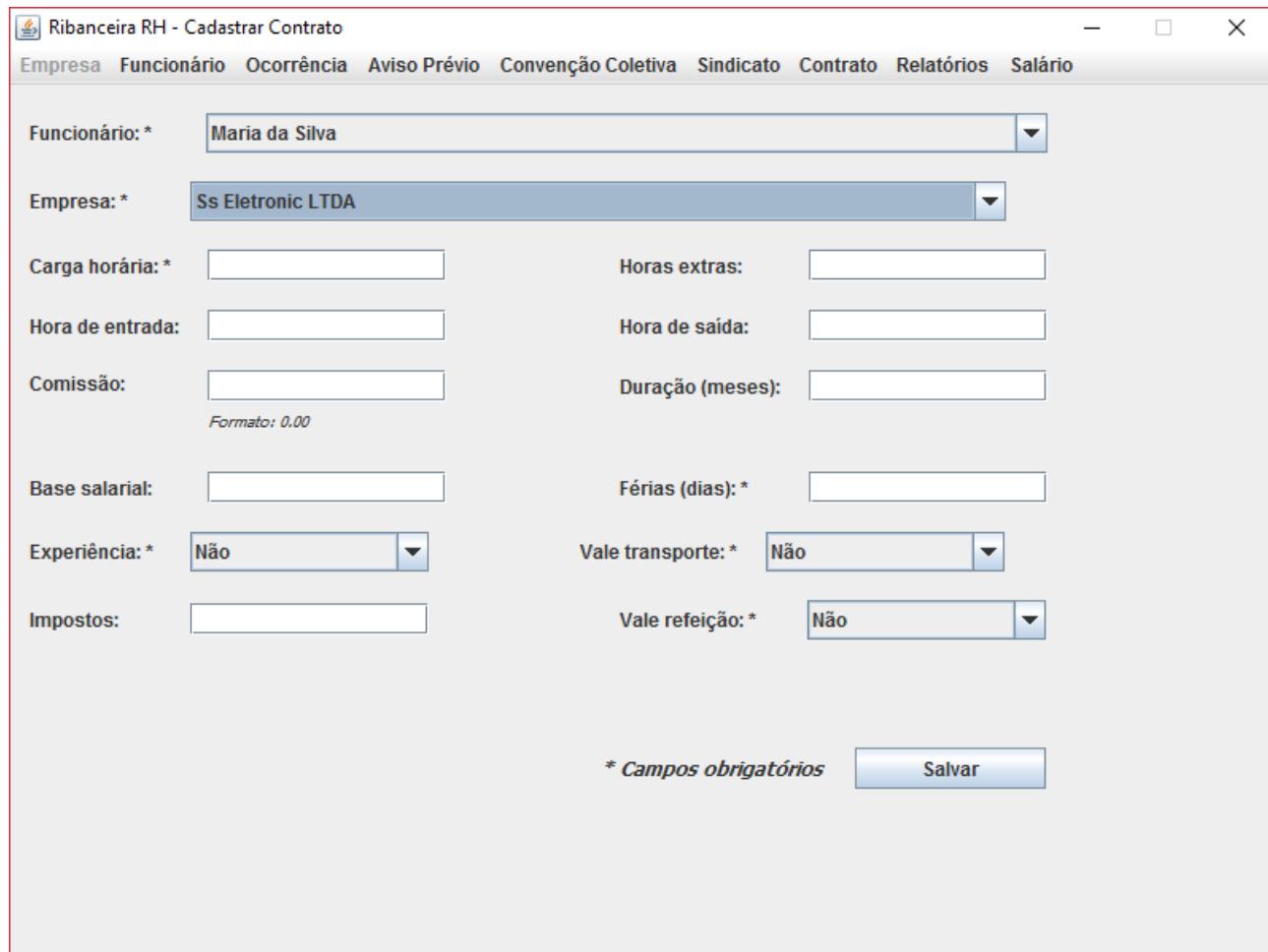
Figura 67 – Tela de login



Figura 68 – Tela principal



Figura 69 – Tela de gerar informes de rendimento



Ribanceira RH - Cadastrar Contrato

Empresa Funcionário Ocorrência Aviso Prévio Convenção Coletiva Sindicato Contrato Relatórios Salário

Funcionário: * Maria da Silva

Empresa: * Ss Eletronic LTDA

Carga horária: * [] Horas extras: []

Hora de entrada: [] Hora de saída: []

Comissão: [] Duração (meses): []
Formato: 0.00

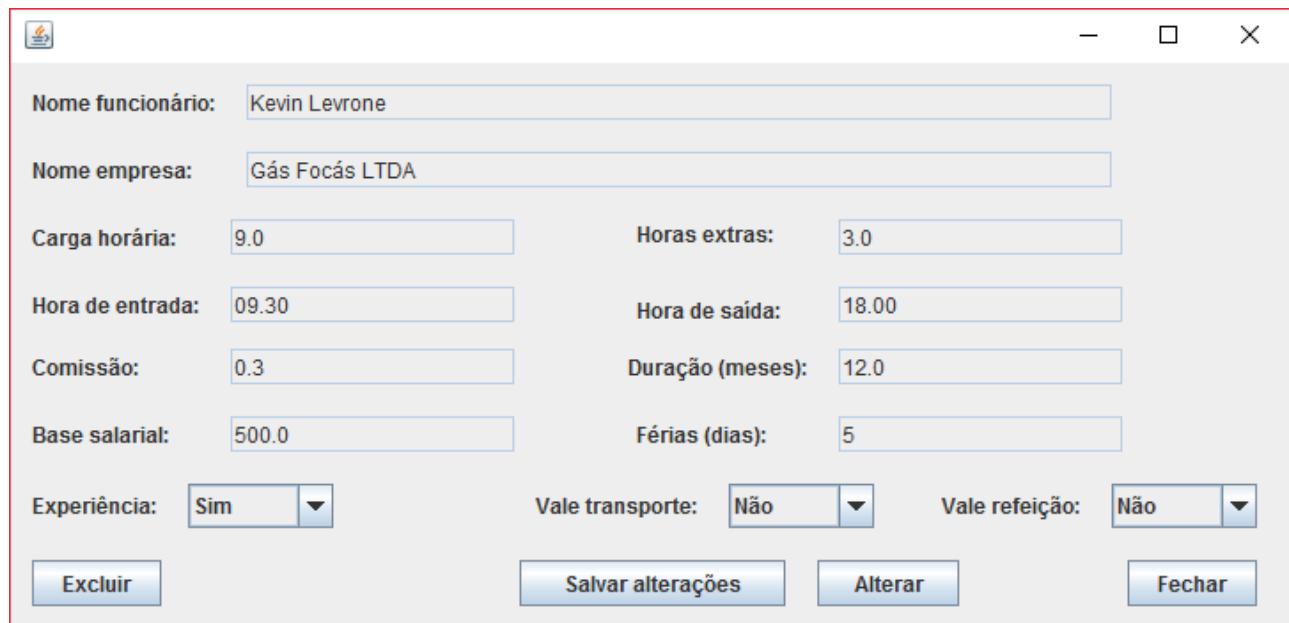
Base salarial: [] Férias (dias): * []

Experiência: * Não Vale transporte: * Não

Impostos: [] Vale refeição: * Não

* Campos obrigatórios Salvar

Figura 70 – Tela de Cadastrar contrato



Nome funcionário: Kevin Levrone

Nome empresa: Gás Focás LTDA

Carga horária: 9.0 Horas extras: 3.0

Hora de entrada: 09.30 Hora de saída: 18.00

Comissão: 0.3 Duração (meses): 12.0

Base salarial: 500.0 Férias (dias): 5

Experiência: Sim Vale transporte: Não Vale refeição: Não

Excluir Salvar alterações Alterar Fechar

Figura 71 – Tela de exibir/editar contrato

The screenshot shows a software application window titled "Ribanceira RH - Consultar Contrato". The window has a standard OS X-style title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with the following items: Empresa, Funcionário, Ocorrência, Aviso Prévio, Convenção Coletiva, Sindicato, Contrato, Relatórios, and Salário. The main content area is a table with three columns: "Código Contrato", "Funcionario", and "Empresa". There are two rows of data:

Código Contrato	Funcionario	Empresa
5	Christian Nakata	Gás Focás LTDA
6	Kevin Levrone	Gás Focás LTDA

Figura 72 – Tela de consultar contrato

Ribanceira RH - Cadastrar Funcionário

Empresa Funcionário Ocorrência Aviso Prévio Convenção Coletiva Sindicato Contrato Relatórios Salário

Nome do funcionário: * [Text input]

Data de nascimento: * [Text input]

Endereço: * [Text input]

Telefone: * [Text input]

E-mail: [Text input]

RG: * [Text input]

CPF: * [Text input]

Título de eleitor: * [Text input]

Estado civil: * [Text input]

Número da carteira de trabalho: * [Text input]

* Campos obrigatórios

Figura 73 – Tela de Cadastrar funcionário

Ribanceira RH - Consultar Funcionário

Empresa Funcionário Ocorrência Aviso Prévio Convenção Coletiva Sindicato Contrato Relatórios Salário

Nome Funcionário	RG
Christian Nakata	4.345.123-2
Joao X	11.123.456-45
José de Souza	5.234.567-9
Kevin Levrone	13.222.444-4
Maria da Silva	7.654.345-8

Selecionar

Figura 74 – Tela de consultar funcionário

The screenshot shows a software window with a red border. In the top right corner are three icons: a minus sign for window control, a square for maximize/minimize, and an 'X' for close. The main area contains a form with the following fields:

Nome do funcionário:	Kevin Levrone
Data de nascimento:	16/02/1996
Endereço:	Rua KLRMS, 1602
Telefone:	(44)3333-1122
E-mail:	k.levrone@gmail.com
RG:	13.222.444-4
CPF:	088.043.203-45
Título de eleitor:	2131231
Estado civil:	Solteiro
Número da carteira de trabalho:	12312313
Ativo:	Sim

Below the form are three buttons: "Alterar" (Change), "Salvar Alterações" (Save Changes), and "Excluir" (Delete).

Figura 75 – Tela de exibir/editar funcionário

Design Preview [JPanelConvencaoColetiva]

Sindicato	Código	Ramo de atividade

Data (última data de atualização):

Proporção do salário mínimo:

Desconto por dia:

Figura 76 – Tela de convenção coletiva

Design Preview [JPanelManterSindicatos]

Buscar Cadastrar Alterar

Nome	Código	Endereço	Telefone

Excluir sindicato Atualizar Lista Alterar Sair

Figura 77 – Tela de sindicato

Visualização do Design [JPanelConsultarAviso]

Avisos registrados:

Data de emissão	Data da Rescisão	Motivo	Justificado	Funcionário
			<input type="checkbox"/>	

[Gerar Arquivo](#) [Editar Aviso](#)

Figura 78 – Tela de consultar aviso

Visualização do Design [JFrameEditarAviso]

Data de emissão do aviso: *

Data da rescisão: *

Funcionário : *

Motivo de aviso:

Justificado

* Campo obrigatório

Figura 79 – Tela de editar aviso

Visualização do Design [JPanelRegistrarAviso]

Data de emissão do aviso:

Data da rescisão:

Funcionário:

Motivo de aviso: Justa causa

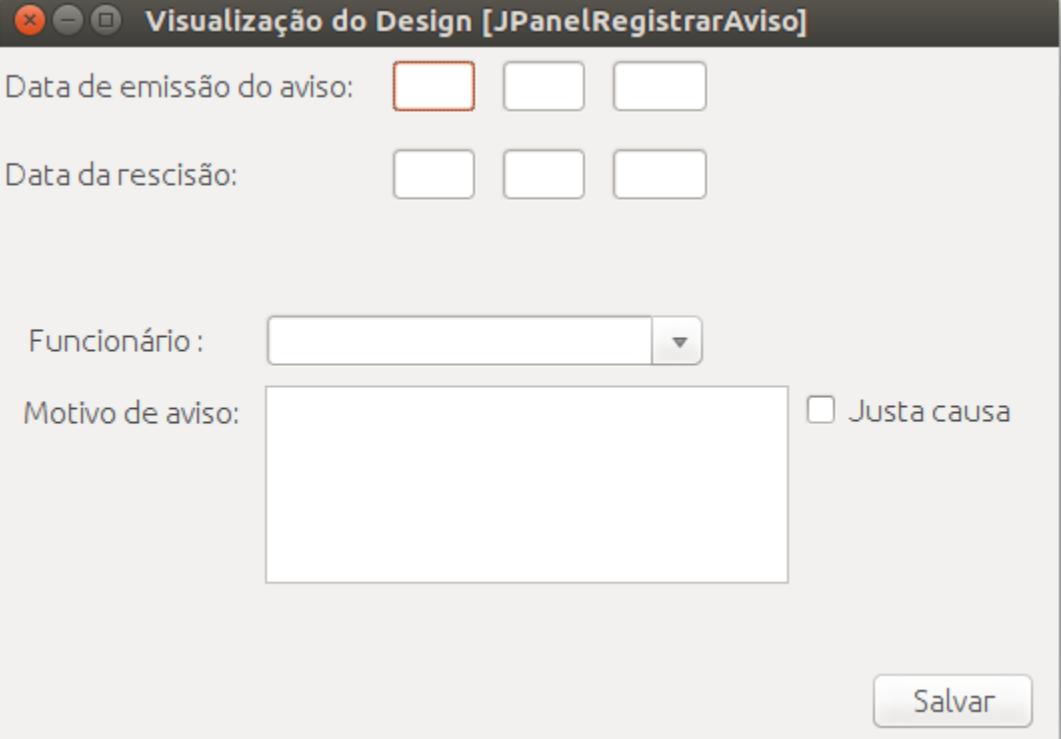


Figura 80 – Tela de cadastrar aviso

Visualização do Design [JFrameEmiteHolerite]

Funcionário:



Figura 81 – Tela de emitir holerite

Visualização do Design [JPanelConsultarOcorrencia]

Lista de Ocorrências:

Funcionário	Ocorrência	Data	Justificada
			<input type="checkbox"/>

Editar ocorrência

Figura 82 – Tela de consultar ocorrência

Visualização do Design [JFrameEditarOcorrencia]

Nome do Funcionário: *

Empresa:

Data da ocorrência: DD/MM/AAAA *

Tipo de ocorrência: * Justificada *

Valor de acréscimo ao salário: * * Campo obrigatório

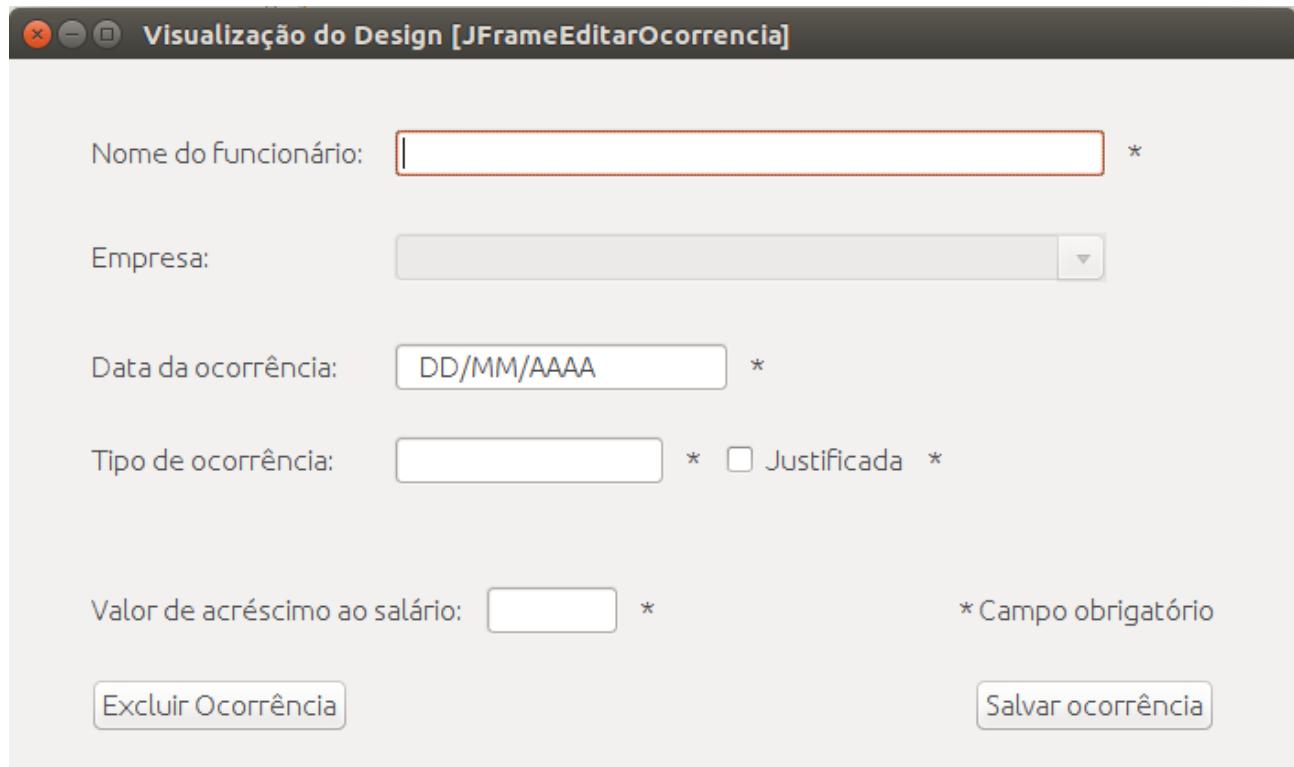


Figura 83 – Tela de editar ocorrência

Relatórios Gerais

Critério do relatório: Ativo Não ativo

Nome empresa:

Critério do funcionário:

Dados do relatório:

<input checked="" type="checkbox"/> Nome	<input type="checkbox"/> Data Nasc.
<input type="checkbox"/> Endereço	<input type="checkbox"/> Telefone
<input type="checkbox"/> E-mail	<input type="checkbox"/> RG
<input type="checkbox"/> CPF	<input type="checkbox"/> Nº Cart. trab.

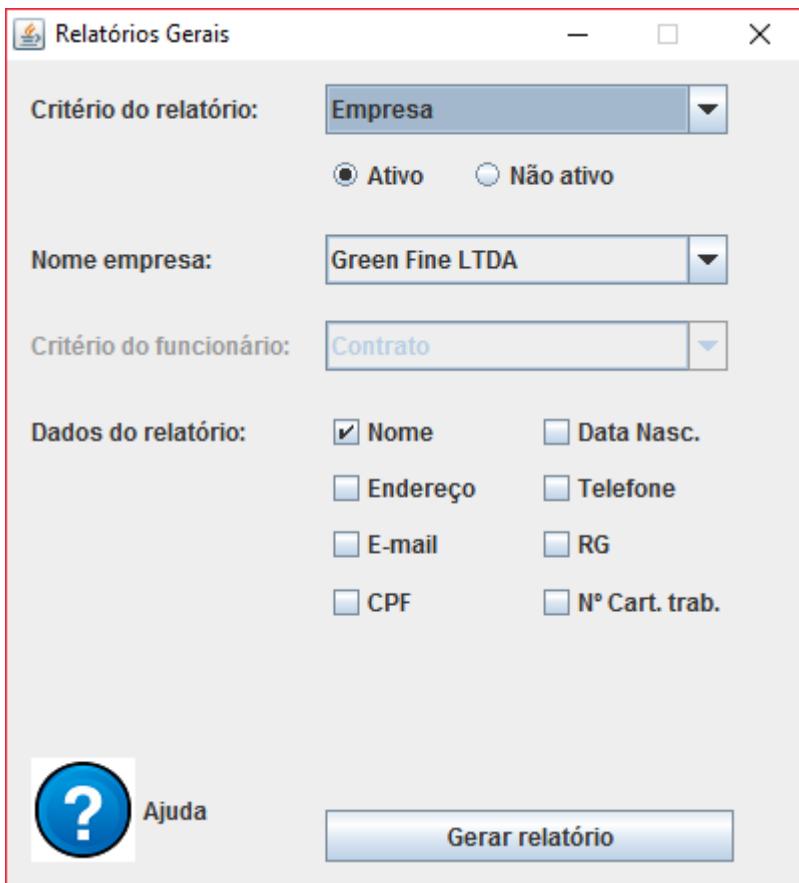


Figura 84 – Tela de relatórios gerais