



**UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I - CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM BACHARELADO EM COMPUTAÇÃO**

LUCAS SANTOS ANDRADE

**PROCESSO DE MAPEAMENTO E DOCUMENTAÇÃO DO SISTEMA DE
SOLICITAÇÃO DE CIRURGIAS – REGNUTES**

CAMPINA GRANDE

2024

LUCAS SANTOS ANDRADE

**PROCESSO DE MAPEAMENTO E DOCUMENTAÇÃO DO SISTEMA DE
SOLICITAÇÃO DE CIRURGIAS – REGNUTES**

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Daniel Scherer.

CAMPINA GRANDE

2024

É expressamente proibido a comercialização deste documento, tanto na forma impressa como eletrônica. Sua reprodução total ou parcial é permitida exclusivamente para fins acadêmicos e científicos, desde que na reprodução figure a identificação do autor, título, instituição e ano do trabalho.

201 08 001 Andrade, Lucas Santos.

Processo de mapeamento e documentação do sistema de solicitação de cirurgias - Reg Nutes [manuscrito] / Lucas Santos Andrade. - 2024.

84 p. : il. colorido.

Digitado. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2024. "Orientação : Prof. Dr. Daniel Scherer, Coordenação do Curso de Computação - CCT. "

1. Mapeamento de software. 2. Documentação de software. 3. Manutenção de software. 4. Modelagem de documentações. 5. Reg Notes. I. Título

21. ed. CDD 005

LUCAS SANTOS ANDRADE

PROCESSO DE MAPEAMENTO E DOCUMENTAÇÃO DO SISTEMA DE
SOLICITAÇÃO DE CIRURGIAS – REGNUTES

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

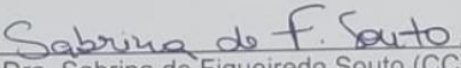
Área de concentração: Engenharia de Software.

Aprovado(a) em: 05 / setembro / 2024.

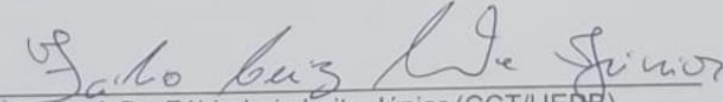
BANCA EXAMINADORA



Prof. Dr. Daniel Scherer (CCT/UEPB)
Orientador(a)



Profa. Dra. Sabrina de Figueiredo Souto (CCT/UEPB)
Examinador(a)



Prof. Dr. Fábio Luiz Leite Júnior (CCT/UEPB)
Examinador(a)

AGRADECIMENTOS

Ao meu pai, o meu herói, por me mostrar persistência e valor na educação e me incentivar a buscar mais, sempre me guiando pelo caminho correto.

À minha mãe, minha rainha, que sempre me mostrou o sentido de gentileza e responsabilidade, mesmo com todas as dificuldades, e me ensinou a demonstrar o meu melhor e ajudar quando puder.

À minha irmã, que buscou me ajudar da melhor forma ao mesmo tempo em que lutava por seus sonhos. Sua hora de alcançá-los está mais próxima do que imagina. Amo muito você.

À professora Sabrina pela oportunidade de me ajudar a crescer. Por meio da sua guia, consegui me ver melhor no curso e também vislumbrar um futuro melhor para minha família graças a esses pequenos passos.

Ao professor Daniel, que me suportou e me orientou de forma paciente e direcionada. Graças ao senhor, finalmente, terminei o grande desafio, do qual tive medo de encarar.

Ao nutes por me mostrar o grande horizonte que é a área de tecnologia. Graças ao grande trabalho de vocês, consigo me ver cada vez mais no meio.

RESUMO

Este trabalho detalha o processo de documentação do sistema RegNUTES, voltado para a solicitação de cirurgias com uso da linguagem UML (Unified Modeling Language). A ênfase está na importância da documentação como ferramenta essencial para a manutenção de software, assegurando que o sistema possa ser compreendido e modificado por desenvolvedores de forma eficiente. O uso de diagramas, como o de casos de uso e atividades, possibilitou uma visão clara das interações e fluxos internos do sistema, permitindo identificar pontos críticos para futuras atualizações e correções. A modelagem estrutural do sistema foi realizada com foco na continuidade do projeto, facilitando a implementação de novas funcionalidades e a correção de erros e minimizando os riscos de falhas e interrupções no funcionamento.

Palavras-Chave: mapeamento de software; documentação de software; manutenção de software; modelagem de documentações.

ABSTRACT

This report presents the documentation process of the RegNotes system, which manages surgery requests, using the Unified Modeling Language (UML). The focus is on the documentation's critical role as a tool for software maintenance, ensuring that developers can easily understand and modify the system. By employing diagrams such as use case and activity diagrams, it became possible to clearly visualize user interactions and internal workflows, identifying key points for future updates and fixes. The system's structural modeling was carried out with an emphasis on project continuity, facilitating the implementation of new features and error corrections, minimizing the risks of failures and system interruptions.

Keywords: software mapping; software documentation; software maintenance; modeling of documentations.

LISTA DE ILUSTRAÇÕES

Figura 1 –	Etapas do processo metodológico	14
Figura 2 –	Exemplo de um diagrama de classes	20
Figura 3 –	Exemplo de um diagrama de objetos	21
Figura 4 –	Exemplo de um diagrama de pacotes	22
Figura 5 –	Exemplo de um diagrama de sequência	23
Figura 6 –	Exemplo de um diagrama de comunicação	24
Figura 7 –	Exemplo de um diagrama de máquina de estados	25
Figura 8 –	Exemplo de um diagrama de componentes	26
Figura 9 –	Exemplo de um diagrama de implantação	27
Figura 10 –	Exemplo de um diagrama de estrutura composta	28
Figura 11 –	Exemplo de um diagrama de tempo	29
Figura 12 –	Exemplo de um diagrama de casos de uso	31
Figura 13 –	Exemplo de atores de um diagrama de casos de uso	32
Figura 14 –	Exemplo de caso de uso	32
Figura 15 –	Exemplo de relação por associação	33
Figura 16 –	Exemplo de relação por generalização	33
Figura 17 –	Exemplo de relação por inclusão	34
Figura 18 –	Exemplo de relação por extensão	34
Figura 19 –	Exemplo de sistema	35
Figura 20 –	Exemplo de um diagrama de atividade	37
Figura 21 –	Exemplo de uma atividade	38
Figura 22 –	Exemplo de um nó de ação	38
Figura 23 –	Exemplo de um fluxo de controle	39
Figura 24 –	Exemplo de uma abordagem com nó inicial	39
Figura 25 –	Exemplo de um nó de final de atividade	39
Figura 26 –	Exemplo de um nó de decisão	40
Figura 27 –	Exemplo de um nó de bifurcação	40
Figura 28 –	Exemplo de um nó de união	41
Figura 29 –	Exemplo de um nó de final de fluxo	41
Figura 30 –	Exemplo de um fluxo de controle	42
Figura 31 –	Exemplo de um conector	43

Figura 32 – Exemplo da ferramenta Draw.io	44
Figura 33 – Mapa do fluxo de cirurgias	45
Figura 34 – Exemplo de cadastro de uma solicitação	45
Figura 35 – Exemplo de formulário com opção de validar solicitação	46
Figura 36 – Exemplo de formulário para informar o parecer técnico	47
Figura 37 – Exemplo de formulário para reservar a cirurgia	47
Figura 38 – Exemplo de agendamento de cirurgia	48
Figura 39 – Exemplo de rejeição de solicitação	49
Figura 40 – Exemplo de formulário para desfecho	49
Figura 41 – Exemplo de formulário para finalizar solicitação	50
Figura 42 – Mapa do fluxo de leitos	51
Figura 43 – Exemplo de formulário para cadastrar solicitação de leito	52
Figura 44 – Exemplo de formulário para validar solicitação de leito	53
Figura 45 – Exemplo de formulário para cadastrar o parecer técnico	53
Figura 46 – Exemplo de formulário para reservar do leito	54
Figura 47 – Exemplo de confirmação de ciência sobre a solicitação do leito	55
Figura 48 – Exemplo de confirmação de chegada do paciente no leito	55
Figura 49 – Exemplo de caso de transferência interna de leito	56
Figura 50 – Exemplo de formulário para desfecho	57
Figura 52 – Mapa do mapeamento da criação da documentação	66
Figura 53 – Diagrama de casos de uso do fluxo de cirurgias	69
Figura 54 – Diagrama de atividade do fluxo de adicionar nova solicitação de cirurgia	73
Figura 55 – Diagrama de casos de uso do fluxo de leitos	76
Figura 56 – Diagrama de atividade do fluxo de adicionar nova solicitação de leito	79

LISTA DE ABREVIATURAS E SIGLAS

UML	Unified Modeling Language
VS CODE	Visual Studio Code

SUMÁRIO

1	INTRODUÇÃO	12
2	METODOLOGIA	14
2.1	Pesquisa bibliográfica	14
2.2	Abordagem para análise de sistemas	15
2.3	Modelagem UML	15
2.4	Escolha das ferramentas	15
2.5	Validação dos diagramas	16
3	REFERENCIAL TEÓRICO	17
3.1	GitHub	17
3.2	Visual Studio Code	17
3.3	UML	18
3.3.1	<i>Diagrama de classe</i>	19
3.3.2	<i>Diagrama de objetos</i>	20
3.3.3	<i>Diagrama de pacotes</i>	21
3.3.4	<i>Diagrama de sequência</i>	22
3.3.5	<i>Diagrama de comunicação</i>	23
3.3.6	<i>Diagrama de máquina Estados</i>	24
3.3.7	<i>Diagrama de componentes</i>	25
3.3.8	<i>Diagrama de implantação</i>	26
3.3.9	<i>Diagrama de estrutura composta</i>	27
3.3.10	<i>Diagrama de tempo/temporização</i>	28
3.4	Diagrama de casos de uso e diagrama de atividade	29
3.4.1	<i>Diagrama de casos de uso</i>	29
3.4.1.1	<i>Atores</i>	31
3.4.1.2	<i>Casos de uso</i>	32
3.4.1.3	<i>Relações</i>	33
3.4.1.4	<i>Sistema</i>	35
3.4.2	<i>Diagrama de atividade</i>	35
3.4.2.1	<i>Atividades</i>	37
3.4.2.2	<i>Nó de ação</i>	38
3.4.2.3	<i>Fluxo de controle</i>	38
3.4.2.4	<i>Nó Inicial</i>	39

3.4.2.5	<i>Nó de final de atividade</i>	39
3.4.2.6	<i>Nó de decisão</i>	40
3.4.2.7	<i>Nó de bifurcação e de união</i>	40
3.4.2.8	<i>Nó de final de fluxo</i>	41
3.4.2.9	<i>Fluxo de controle</i>	42
3.4.2.10	<i>Conectores</i>	43
3.5	Draw.io	43
3.6	RegNUTES	44
3.6.1	Cirurgias	44
3.6.1.1	<i>Cadastro da solicitação</i>	45
3.6.1.2	<i>Validação e parecer médico</i>	46
3.6.1.3	<i>Reserva da solicitação</i>	47
3.6.1.4	<i>Agendamento da cirurgia</i>	48
3.6.1.5	<i>Rejeição</i>	48
3.6.1.6	<i>Desfecho da solicitação</i>	49
3.6.1.7	<i>Finalizar solicitação</i>	50
3.6.2	Leitos	50
3.6.2.1	<i>Cadastro da solicitação</i>	51
3.6.2.2	<i>Validação e parecer médico</i>	52
3.6.2.3	<i>Reserva da solicitação</i>	54
3.6.2.4	<i>Ciência e confirmada a chegada</i>	55
3.6.2.5	<i>Transferência interna</i>	56
3.6.2.6	<i>Confirmação de desfecho</i>	56
3.6.3	Outros fluxos	57
3.6.3.1	<i>Cadastro de usuários</i>	57
3.6.3.2	<i>Cadastro de unidades de saúde</i>	59
3.6.3.3	<i>Autorização do uso das cirurgias</i>	59
3.6.3.4	<i>Estatísticas de cirurgias e leitos</i>	60
4	DESENVOLVIMENTO	61
5	RESULTADOS E DISCUSSÕES	65
5.1	Mapeamento e construção	65
5.1.1	Código fonte e comentários de código	66
5.1.2	Manual do usuário e helpdesk	67
5.1.3	Diagrama de atividade parcial	67

5.1.4	<i>Diagrama de casos de uso</i>	68
5.1.5	<i>Diagrama de atividade final</i>	68
5.2	Cirurgias	68
5.2.1	<i>Diagrama de Casos de Uso: Cirurgias</i>	69
5.2.1.1	<i>Descrição Geral do Diagrama</i>	70
5.2.1.2	<i>Atores Envolvidos</i>	70
5.2.1.3	<i>Principais Casos de Uso</i>	71
5.2.1.4	<i>Interpretação e Impacto no Projeto</i>	71
5.2.1.5	<i>Discussão</i>	72
5.2.2	<i>Diagrama de Atividade: Cirurgias</i>	72
5.2.2.1	<i>Descrição Geral do Diagrama</i>	73
5.2.2.2	<i>Atores Envolvidos</i>	73
5.2.2.3	<i>Principais Decisões e Alternativas</i>	74
5.2.2.4	<i>Interpretação e Impacto no Projeto</i>	74
5.2.2.5	<i>Discussão</i>	74
5.3	Leitos	75
5.3.1	<i>Diagrama de Casos de Uso: Leitos</i>	75
5.3.1.1	<i>Descrição Geral do Diagrama</i>	76
5.3.1.2	<i>Atores Envolvidos</i>	76
5.3.1.3	<i>Principais Casos de Uso</i>	77
5.3.1.4	<i>Interpretação e Impacto no Projeto</i>	78
5.3.1.5	<i>Discussão</i>	78
5.3.1	<i>Diagrama de Atividade: Leitos</i>	78
3.6.3.1	<i>Descrição Geral do Diagrama</i>	79
3.6.3.2	<i>Atores Envolvidos</i>	79
3.6.3.3	<i>Principais Decisões e Alternativas</i>	79
3.6.3.4	<i>Interpretação e Impacto no Projeto</i>	80
3.6.3.5	<i>Discussão</i>	80
6	CONCLUSÃO	81
	REFERÊNCIAS	83

1 INTRODUÇÃO

O gerenciamento de recursos em unidades de saúde é um processo complexo e essencial para o bom funcionamento do sistema público de saúde. A alocação de leitos e o agendamento de cirurgias, por exemplo, exigem uma coordenação precisa e eficiente, que garanta o atendimento adequado aos pacientes e a otimização dos recursos disponíveis. Nesse contexto, sistemas informatizados têm se tornado ferramentas indispensáveis para auxiliar na gestão desses processos, proporcionando maior agilidade, segurança e controle nas operações hospitalares.

O desenvolvimento do sistema RegNUTES, foco deste trabalho, surge da necessidade de melhorar a gestão de leitos e cirurgias em unidades de saúde, buscando minimizar falhas operacionais e otimizar a comunicação entre os profissionais envolvidos. A ausência de uma ferramenta centralizada capaz de gerenciar essas atividades de forma integrada e padronizada foi um dos principais problemas identificados, motivando a criação de uma solução tecnológica que atendesse a essas demandas específicas.

A escolha do RegNUTES para este trabalho foi motivada por sua complexa arquitetura baseada em micros serviços, que proporciona flexibilidade e escalabilidade, mas também impõe desafios significativos de coordenação entre serviços independentes. Além disso, o sistema é responsivo, o que garante acessibilidade em diferentes dispositivos e contextos, exigindo atenção especial ao design de interface e à interação entre *front-end* e *back-end*. O RegNUTES também depende de uma constante interação com os usuários para se manter funcional, já que dados e solicitações em tempo real são essenciais para seu correto funcionamento. Esses aspectos técnicos exigem uma documentação sólida e detalhada, especialmente em termos de integração entre serviços, fluxos de usuários e manutenção contínua, o que torna o sistema um exemplo ideal para a aplicação de técnicas de modelagem como a UML.

Com base nessas premissas, o presente trabalho aborda as etapas de desenvolvimento da documentação do RegNUTES, passando pela pesquisa bibliográfica, análise do sistema existente, modelagem de diagramas de caso de uso e atividade e validação das representações junto aos desenvolvedores. A partir dessa abordagem, espera-se que o RegNUTES possa ser compreendido e aprimorado de forma mais eficiente por futuras equipes de desenvolvimento.

Além disso, é discutido o processo de modelagem de diagramas, como os de casos de uso e de atividades, ressaltando sua importância para a visualização das interações

entre os atores do sistema e o fluxo de atividades críticas, como a solicitação de leitos e cirurgias. Esses diagramas, ao detalhar as responsabilidades dos atores e os processos internos do sistema, garantem uma comunicação clara e eficiente entre os diversos profissionais envolvidos no projeto.

Esse trabalho se divide em 6 seções, sendo que na seção 1, é apresentada a introdução, que contextualiza o tema e a motivação para o desenvolvimento do projeto. A seção 2 aborda a fundamentação teórica, detalhando os principais conceitos e referências utilizados no estudo. A seção 3 explora a metodologia aplicada, descrevendo as etapas e os métodos utilizados para a criação e modelagem do sistema. Na seção 4, são discutidos os resultados obtidos com a modelagem dos diagramas de Caso de Uso e Atividade. Finalmente, a seção 5 apresenta as considerações finais, sugerindo também possíveis direções para trabalhos futuros.

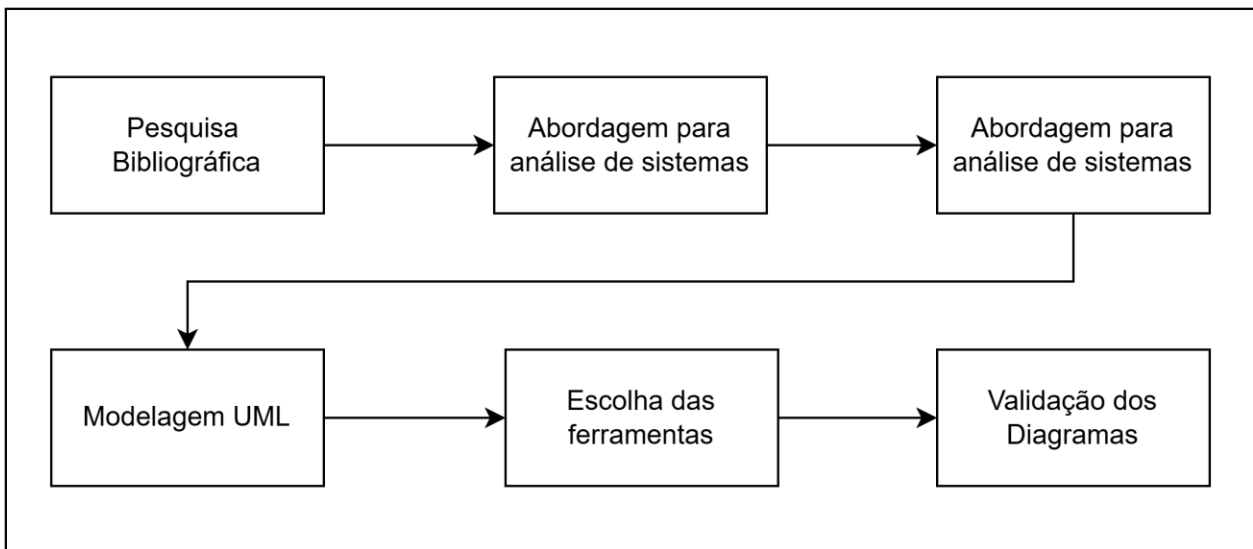
2 METODOLOGIA

A metodologia utilizada para a criação da documentação do RegNUTES foi baseada em princípios de engenharia de software, com ênfase na modelagem de processos e com uso da linguagem UML.

O objetivo é garantir que o sistema seja representado de forma clara e compreensível, facilitando a manutenção e a implementação de novas funcionalidades pelos desenvolvedores.

O processo metodológico foi dividido nas seguintes etapas: pesquisa bibliográfica, análise de sistemas, modelagem UML, Escolha das ferramentas e validação das representações, como pode ser visto na figura 1.

Figura 1 – Etapas do processo metodológico



Fonte: Elaborado pelo autor, 2024.

2.1 Pesquisa bibliográfica

A primeira etapa da metodologia consiste em uma ampla pesquisa bibliográfica. Nela, foram selecionados livros, artigos acadêmicos e guias de boas práticas relacionados à documentação de sistemas, modelagem de software e engenharia de requisitos. O foco principal é o uso da linguagem UML, escolhida por sua padronização internacional e aplicabilidade em diversos tipos de sistemas.

Os tópicos abordados na pesquisa incluíram: a) engenharia de software, para compreender como organizar a documentação de sistemas; b) UML, com estudo dos diferentes diagramas, suas finalidades e melhores práticas para aplicação; e c)

modelagem de casos de uso e atividades, a fim de garantir que esses diagramas sejam adequados às necessidades do RegNUTES.

Esses materiais forneceram a base teórica necessária para conduzir o processo de documentação e auxiliaram na escolha dos diagramas mais adequados para representar o sistema.

2.2 Abordagem para análise de sistemas

A análise do sistema RegNUTES foi conduzida com o objetivo de entender seu funcionamento e mapear suas funcionalidades. Como o sistema já está em operação, a análise se concentrou em examinar a documentação existente, como o manual do usuário, por exemplo, e investigar o código-fonte para entender os processos internos.

Essa abordagem foi escolhida para mitigar a ausência de uma documentação detalhada prévia, utilizando as informações já disponíveis e complementando-as com a análise direta do código. O uso de técnicas de engenharia reversa foi considerado adequado para mapear o comportamento do sistema e identificar os principais casos de uso e processos de atividades.

2.3 Modelagem UML

Com base nos princípios da UML, dois tipos de diagramas foram selecionados: o diagrama de casos de uso, escolhido para descrever as interações entre os usuários e o sistema, destacando as permissões e responsabilidades de cada ator envolvido, e o diagrama de atividades, utilizado para detalhar os fluxos de processos, incluindo decisões, exceções e validações.

A escolha desses diagramas foi baseada na necessidade de representar as interações dos usuários com o sistema e detalhar os fluxos de atividades críticas para o funcionamento do RegNUTES.

2.4 Escolha das ferramentas

Para a modelagem dos diagramas, foram utilizadas ferramentas de software compatíveis com a UML, como o Draw.io, por exemplo, por sua facilidade de uso e

capacidade de criar representações visuais padronizadas. Além disso, editores de código foram utilizados para a análise das funcionalidades internas do sistema.

Essas ferramentas foram escolhidas por oferecerem recursos que garantem a padronização e clareza das representações.

2.5 Validação dos diagramas

Após a criação dos diagramas, o processo de validação foi conduzido em colaboração com desenvolvedores familiarizados com o RegNUTES. Esses profissionais forneceram *feedback* quanto à precisão e à adequação dos diagramas, permitindo ajustes e refinamentos para garantir que as representações fossem consistentes com a realidade do sistema.

3 REFERENCIAL TEÓRICO

Para que haja uma compreensão sobre o assunto, faz-se necessária, inicialmente, uma apresentação básica sobre as ferramentas que foram trabalhadas no decorrer do processo e que serviram para coleta e teste de comportamento dos materiais que foram usados na modelagem, utilizando o Visual Studio Code e o Github.

É importante, ainda, um estudo sobre o conjunto de regras e padrões usados para que o projeto tivesse uma base estruturada e conhecida, facilitando, assim, a compreensão dos diagramas. No caso do presente trabalho, esse processo foi facilitado através do uso da UML.

Por último, também é imperioso descrever os softwares utilizados, dentre os quais o Draw.io, útil para desenhar os diagramas, e o sistema de solicitação de cirurgias RegNUTES, referência para esse material.

3.1 GitHub

O GitHub é uma plataforma de hospedagem de código-fonte que facilita a colaboração e o versionamento de projetos de software, sendo amplamente utilizada por desenvolvedores em todo o mundo (GitHub, 2024). Ela facilita a gestão de mudanças em projetos de software, tanto para desenvolvedores individuais quanto para equipes distribuídas.

Uma das principais características do GitHub é sua interface web, a qual permite a interação com repositórios e a realização de operações como *commit*, *merge* e *pull requests*. A integração com ferramentas, dentre as quais GitHub Desktop, IDEs e Visual Studio Code, também facilita o envio de código.

3.2 Visual Studio Code

O Visual Studio Code (VS Code) é um editor de código-fonte amplamente utilizado, oferecendo suporte para várias linguagens de programação e integração com ferramentas de versionamento, como o GitHub (Microsoft, 2024).

O VS Code destaca-se por seu suporte a uma grande quantidade de linguagens de programação e extensões, adaptando-se a diferentes projetos, como desenvolvimento web e programação em linguagens de baixo nível.

3.3 UML

A *Unified Modeling Language* (UML) ou Linguagem de Modelagem Unificada é uma linguagem de modelagem utilizada para representar visualmente o design de sistemas orientados a objetos, auxiliando no desenvolvimento e documentação de software (OMG, 2021).

É importante pontuar que a UML não é uma linguagem de programação, mas de modelagem, ou seja, consiste em um conjunto de regras que tem o objetivo de auxiliar os engenheiros de software a definir as características do sistema.

Em síntese, esse recurso supre a necessidade de padronizar documentações e encontrar denominadores comuns para facilitar qualquer fase do processo de desenvolvimento de um projeto, tendo por base a simplificação de atividades, como o mapeamento de requisitos, a análise em diagnóstico de possíveis riscos e erros e a catalogação de problemas a serem corrigidos. Do mesmo modo, pode também ser usada para a documentação de estruturação do projeto, comentando sobre suas necessidades, custos, informações mais pertinentes, entre outros dados.

Ressalte-se que a UML é composta por muitos diagramas, posto que seu objetivo é fornecer múltiplas visões do sistema a ser modelado, analisando-o e modelando-o sob diversos aspectos. Dessa forma, ela possibilita a criação de diferentes diagramas que auxiliam no entendimento das interações entre componentes, facilitando a comunicação entre equipes e permitindo a detecção precoce de falhas no sistema (Fowler, 2004).

Da mesma maneira, os diagramas também permitem analisar o sistema em diferentes níveis, podendo trazer foco para a organização estrutural do sistema, para o comportamento de um processo específico, para a definição de um determinado algoritmo ou, até mesmo, para as necessidades físicas da implantação do sistema.

Assim, pode-se dizer que cada diagrama da UML analisa o sistema, ou parte dele, sob uma determinada óptica. É como se o sistema fosse modelado em camadas, sendo que alguns diagramas o analisam de forma genérica, apresentado uma visão externa, enquanto outros oferecem uma visão de camadas mais profundas do software, apresentando uma abordagem mais técnica ou, ainda, visualizando apenas uma característica específica ou um determinado processo do sistema. Em razão disso, o uso de diversos diagramas permite que falhas sejam descobertas, diminuindo a possibilidade de erros futuros.

De tal modo, é possível notar as abrangências que a UML tenta buscar através dos diagramas, trazendo parâmetros diferentes de como um sistema deve se comportar para alcançar melhores resultados em seu trabalho.

Porém, para compreender essas questões de forma mais clara, é preciso analisar o funcionamento de cada um deles, ainda que brevemente. Portanto, nos tópicos que se seguem, foram abordadas suas definições e utilidades a fim de contextualizá-los.

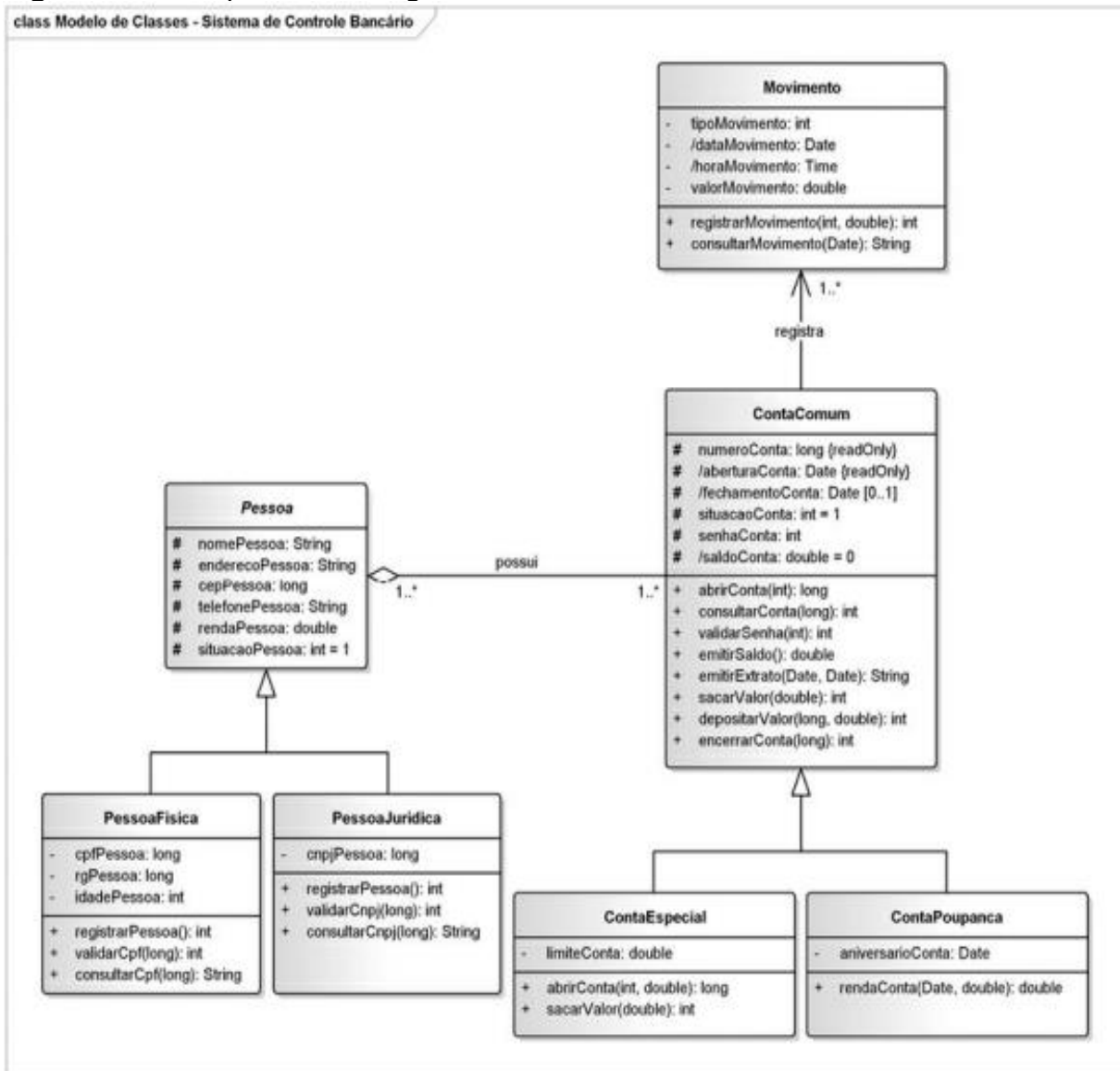
3.3.1 Diagrama de classe

Segundo Gilleanes (2011, p. 34), “um diagrama de classe é utilizado para mostrar a estrutura estática de um sistema, representando suas classes, atributos, métodos e os relacionamentos entre as classes”.

Assim, como se extrai de sua própria denominação, o diagrama de classe define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem, além de estabelecer como as classes se relacionam e trocam informações entre si.

Com isso, essa ferramenta serve de apoio para a maioria dos demais diagramas que possui interação com orientação a objeto. Sua representação pode ser vista conforme na Figura 2.

Figura 2 – Exemplo de um diagrama de classes

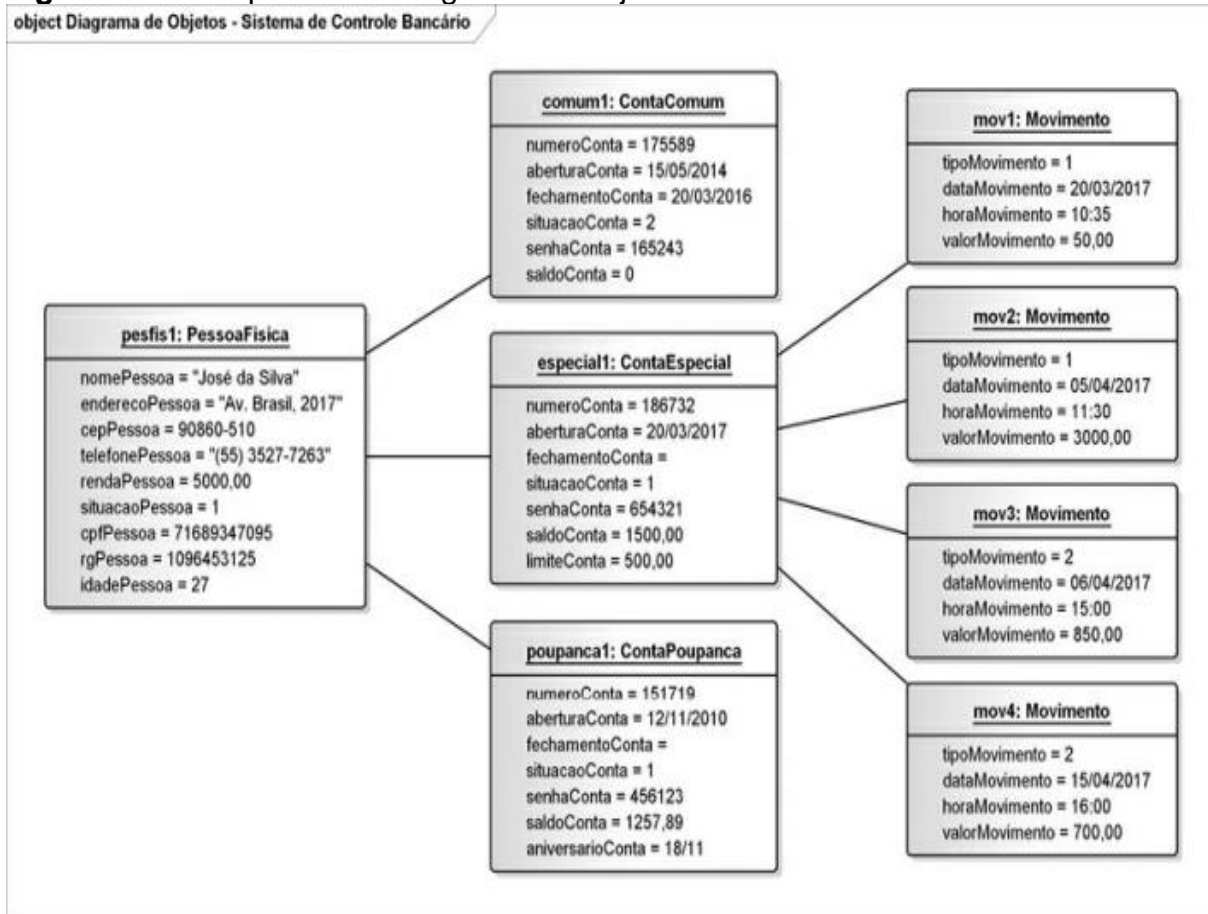


Fonte: Gilleanes, 2011.

3.3.2 Diagrama de objetos

Conforme Gilleanes (2011, p. 35), “um diagrama de objetos representa instâncias concretas das classes em um sistema em um momento específico, mostrando os valores dos atributos e as relações entre esses objetos”.

Significa, portanto, que essa ferramenta demonstra os valores atuais dos atributos dos objetos e as relações entre eles em um momento específico durante a execução de um software, tornando mais fácil a visualização do processo e do desenvolvimento do fluxo do sistema, a exemplo do representado na Figura 3.

Figura 3 – Exemplo de um diagrama de objetos

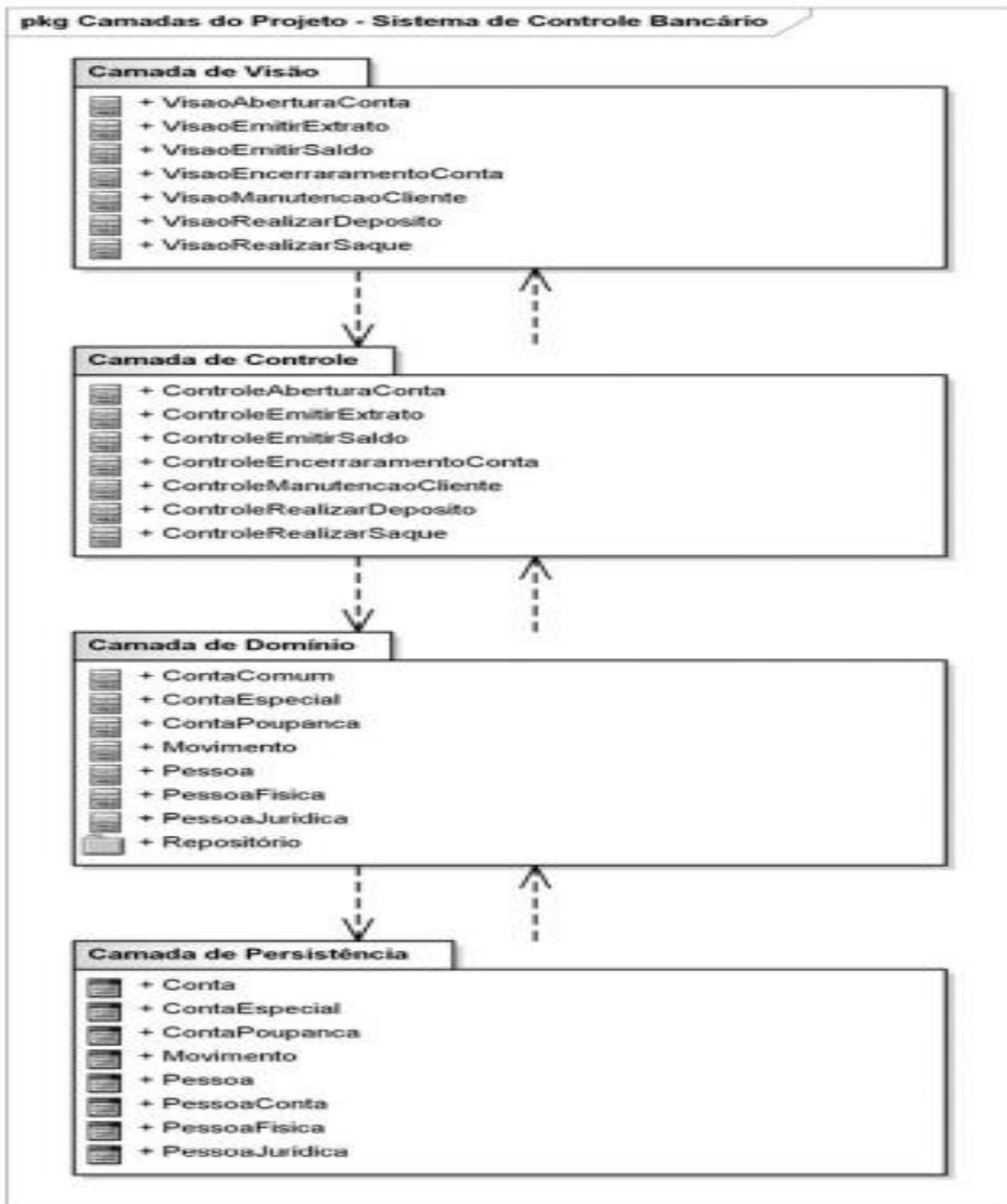
Fonte: Gilleanes, 2011.

3.3.3 Diagrama de pacotes

Para Gilleanes (2011, p. 35), “o diagrama de pacotes é um diagrama estrutural que tem por objetivo representar como os elementos do modelo estão divididos logicamente”. Tais elementos podem ser considerados subsistemas ou submódulos englobados por um sistema de forma a determinar as partes que o compõem.

Logo, o diagrama de pacotes pode ser utilizado de maneira independente ou associado com outros diagramas. Além disso, pode ser utilizado também para definir as camadas de um software ou de um processo de desenvolvimento, bem como para auxiliar na demonstração da arquitetura de uma linguagem, tal como ocorre com a própria UML, indicando, assim, como esses pacotes se relacionam e dependem uns dos outros, oferecendo uma visão geral da estrutura e modularidade do sistema. Para uma melhor noção, segue o exemplo na Figura 4.

Figura 4 – Exemplo de um diagrama de pacotes



Fonte: Gilleanes, 2011.

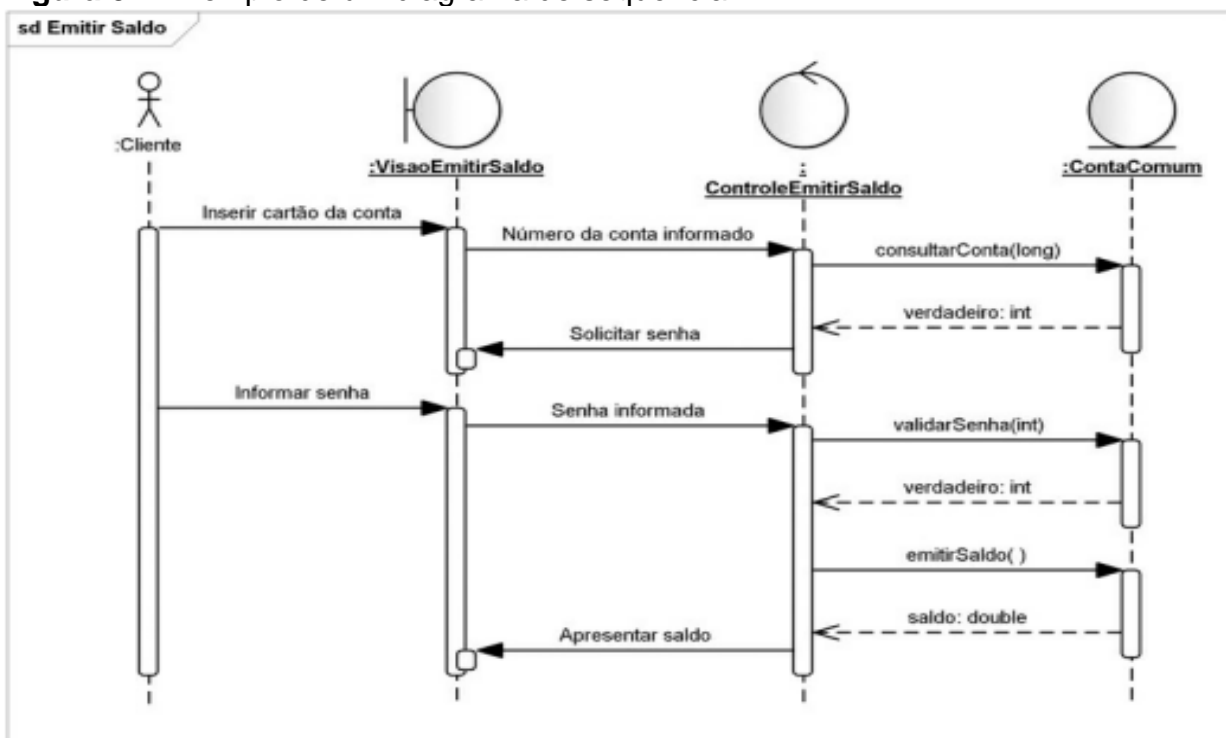
3.3.4 Diagrama de sequência

Ao considerar um diagrama de sequência, é importante compreender sua função, que consiste em realizar uma análise detalhada do fluxo de um caso de uso, focando na

ordem dos eventos e se apoiando no diagrama de classe para determinar o objeto das classes envolvidas em um processo (Gilleanes, 2011).

Um diagrama de sequência identifica o evento gerador do processo modelado e o ator responsável por esse evento. Além disso, determina como o processo deve se desenrolar e ser concluído por meio da chamada de métodos disparados através de mensagens enviadas entre os objetos. Na Figura 5 é possível visualizar um exemplo do diagrama.

Figura 5 – Exemplo de um diagrama de sequência



Fonte: Gilleanes, 2011.

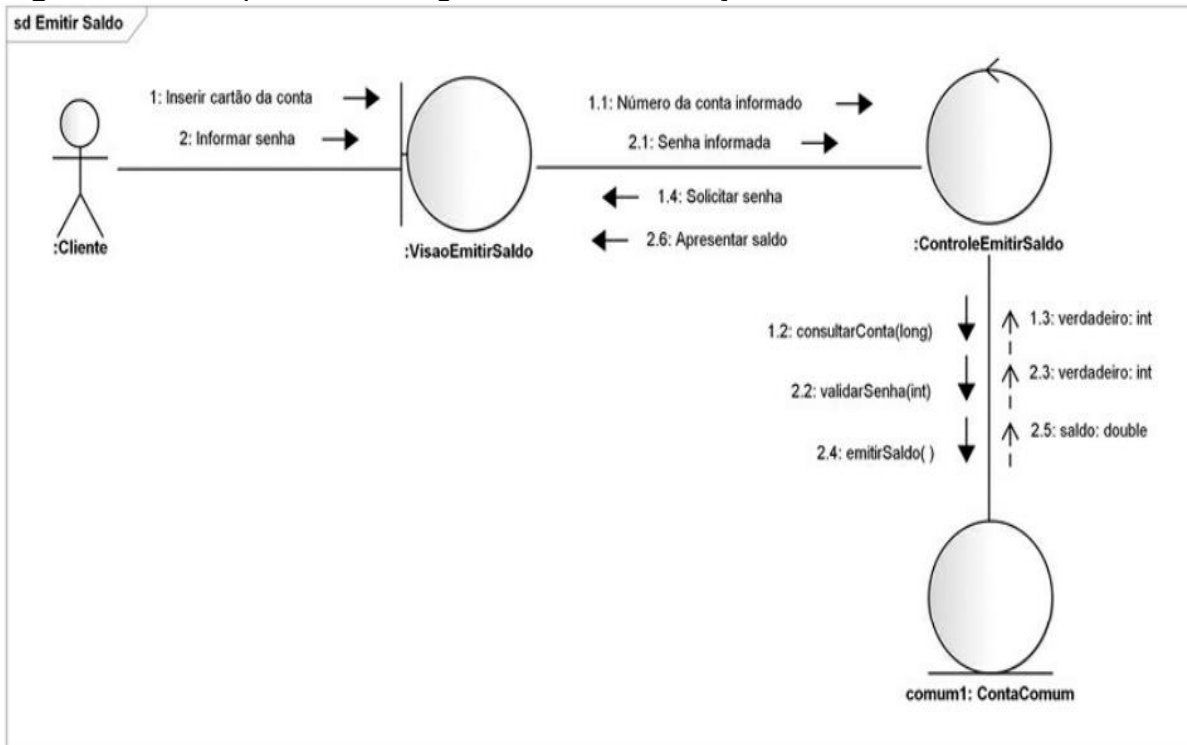
3.3.5 Diagrama de comunicação

Segundo Gilleanes (2011), diagrama de comunicação está amplamente associado ao diagrama de sequência, posto que uma complementa o outro. Embora as informações apresentadas em ambos os diagramas sejam frequentemente semelhantes, o diagrama de comunicação adota um enfoque distinto.

Assim, enquanto o diagrama de sequência foca na temporalidade dos eventos, o diagrama de comunicação concentra-se na organização dos elementos e nas mensagens trocadas entre eles durante o processo. Por isso, esse diagrama oferece uma visão

detalhada de como os componentes interagem, sem considerar a ordem temporal dos eventos. Na Figura 6, é possível analisar a construção desse diagrama.

Figura 6 – Exemplo de um diagrama de comunicação



Fonte: Gilleanes, 2011.

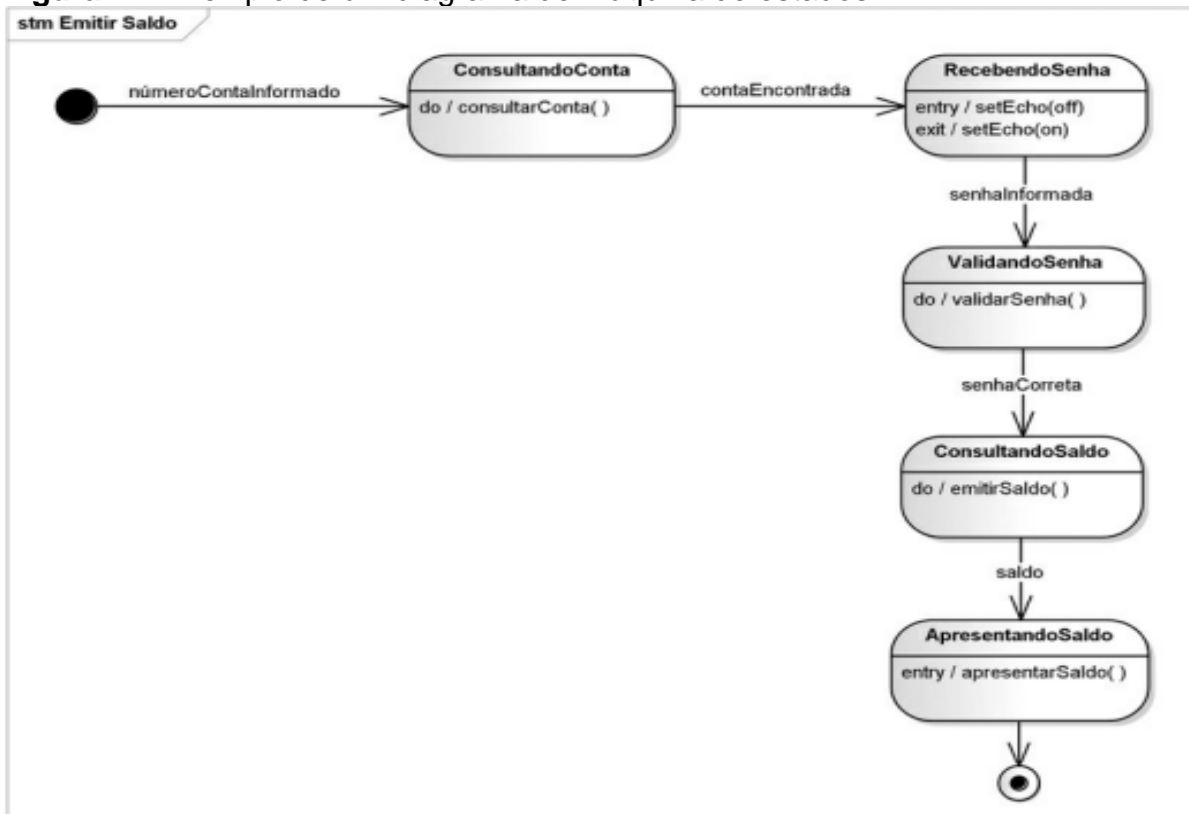
3.3.6 Diagrama de máquina estados

Conforme Gilleanes (2011, p. 38), “o diagrama de máquina de estados demonstra o comportamento de um elemento por meio de conjunto finito de transições de estado”.

Além de ser utilizado para expressar o comportamento de uma parte do sistema, quando é chamado de máquina de estado comportamental, essa ferramenta também serve para expressar o protocolo de uso de parte de um sistema, identificando uma máquina de estado de protocolo.

Assim como o diagrama de sequência, o de máquina de estados pode basear-se em um caso de uso, mas também pode ser utilizado para acompanhar os estados de outros elementos, como, por exemplo, uma instância de uma classe. O exemplo trazido na Figura 7 mostra a aplicação desse diagrama.

Figura 7 – Exemplo de um diagrama de máquina de estados



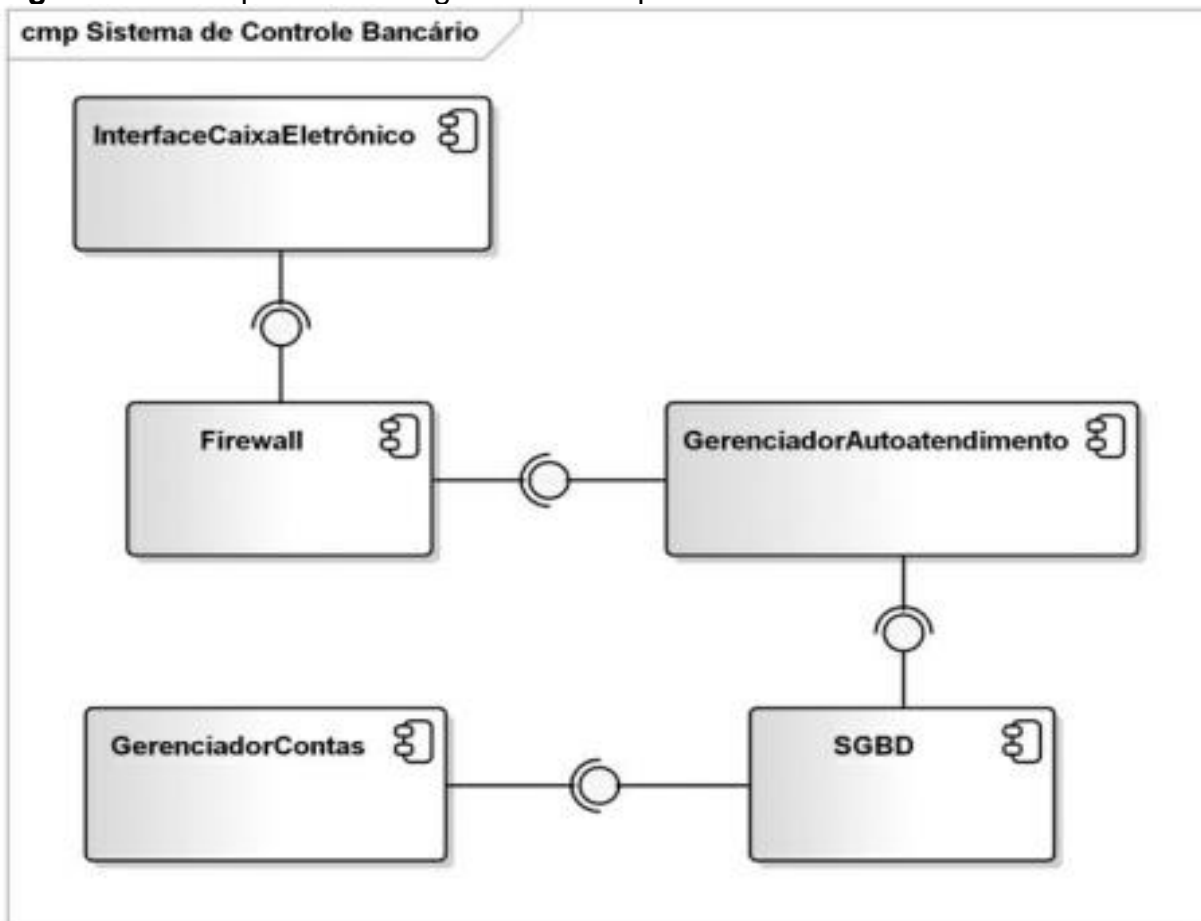
Fonte: Gilleanes, 2011.

3.3.7 Diagrama de componentes

Explica Gilleanes (2011, p. 42) que o diagrama de componentes “é aquele que descreve a organização e a dependência entre os componentes de um sistema”. Significa, portanto, que um componente pode representar tanto um componente lógico, ou seja, de negócio ou de processo, quanto um componente físico, como arquivos contendo código-fonte, arquivos de ajuda (help), bibliotecas, arquivos executáveis, entre outros.

É o diagrama de componentes, portanto, que determina como tais componentes estarão estruturados e interagirão para que o sistema funcione de maneira adequada. Para uma melhor compreensão, segue exemplo na Figura 8.

Figura 8 – Exemplo de um diagrama de componentes



Fonte: Gilleanes, 2011.

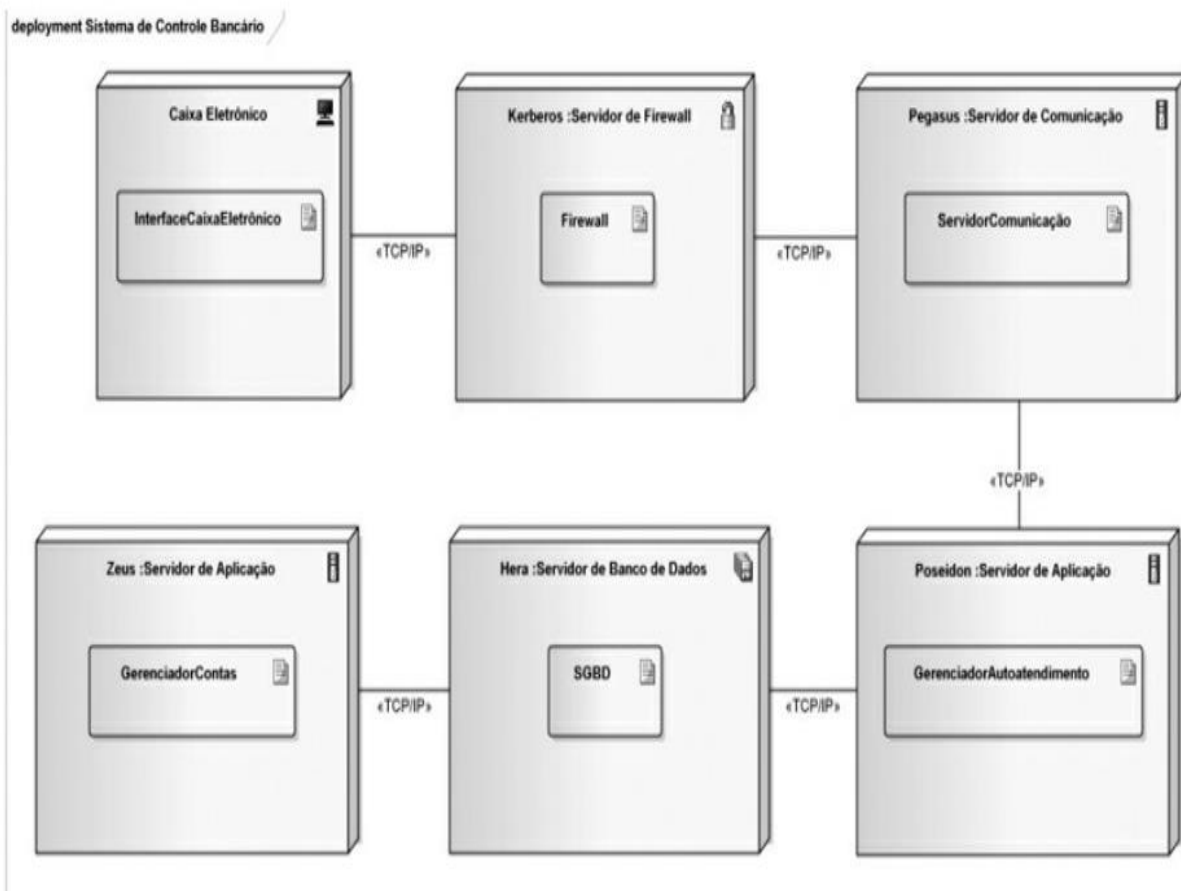
3.3.8 Diagrama de implantação

Segundo Gilleanes (2011, p. 42):

O diagrama de implantação determina as necessidades de hardware do sistema, as características físicas como servidores, estações, topologias e protocolos de comunicação, ou seja, todo o aparato físico sobre o qual o sistema deverá ser executado. Esse diagrama permite demonstrar também como se dará a distribuição dos módulos do sistema, em situações em que estes forem ser executados em mais de um servidor.

O diagrama de implantação, assim, é útil para compreender a arquitetura física do sistema e para o planejamento e manutenção da implantação do software. Um exemplo básico de como ele se desenvolve pode ser visualizado na Figura 9.

Figura 9 – Exemplo de um diagrama de implantação



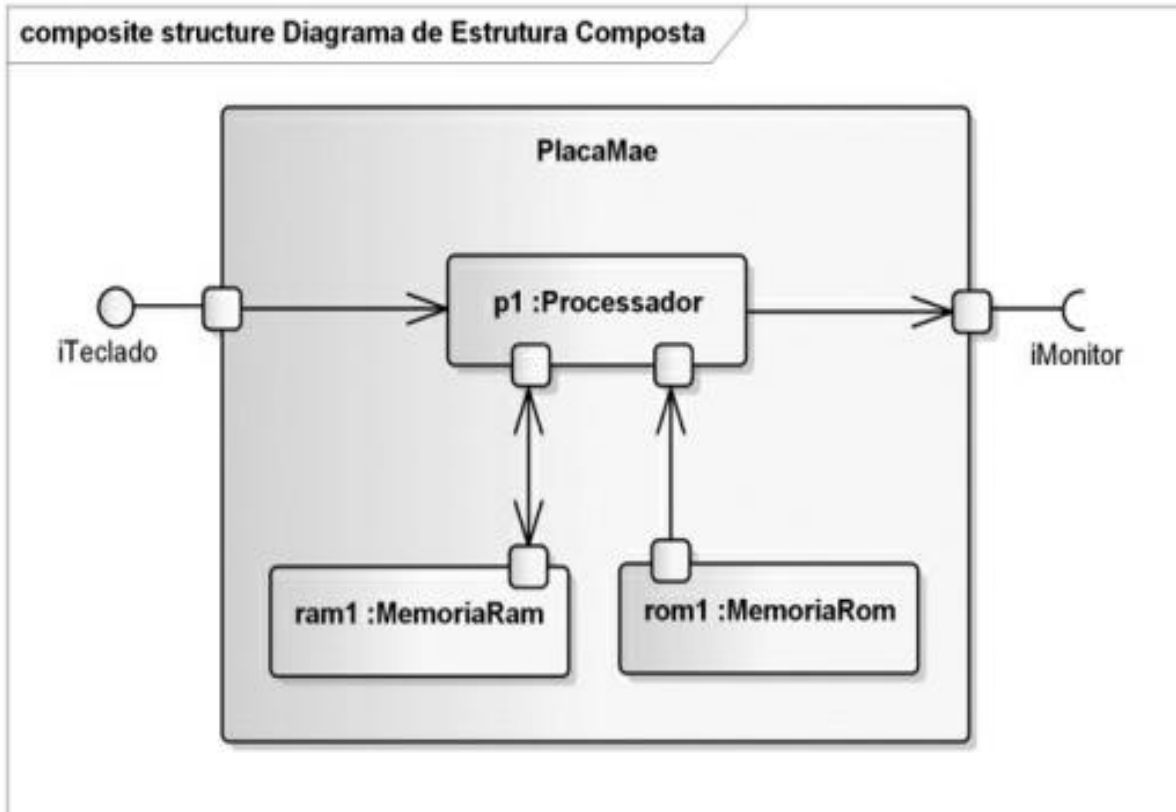
Fonte: Gilleanes, 2011.

3.3.9 Diagrama de estrutura composta

O diagrama de estrutura composta, conforme conceito estabelecido por Gilleanes (2011), consiste em uma descrição da estrutura interna de um classificador, como uma classe ou componente, pormenorizando seus componentes internos e indicando como estes se comunicam e colaboram entre si.

Esse diagrama também é utilizado para descrever uma colaboração em que um conjunto de instâncias cooperam entre si para realizar uma tarefa. É possível afirmar, assim, que ele é útil para entender e documentar a organização e as interações internas de um sistema. Um exemplo dessa ferramenta pode ser encontrado na Figura 10.

Figura 10 – Exemplo de um diagrama de estrutura composta



Fonte: Gilleanes, 2011.

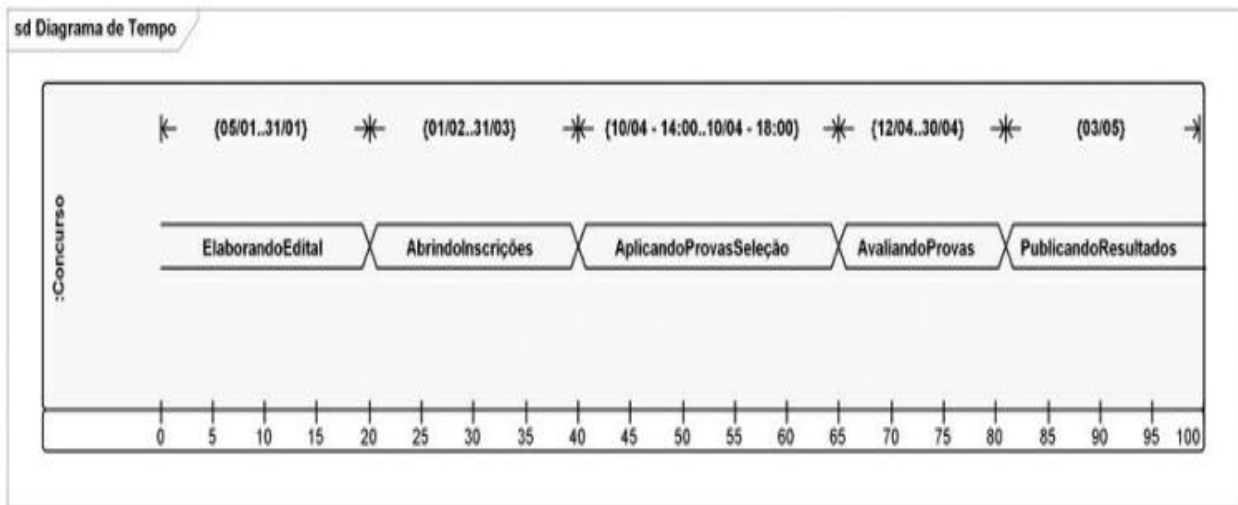
3.3.10 Diagrama de tempo/temporização

Gilleanes (2011, p. 44) explica o diagrama de tempo ou temporização como sendo:

[...] uma técnica que ilustra a progressão do sistema ao longo do tempo, evidenciando acontecimentos, transições de estado e relações temporais entre elementos. Ele é eficaz para compreender a dinâmica temporal do sistema e para avaliar a eficiência e a coordenação das partes.

Para uma melhor compreensão deste conceito, a Figura 11 traz um exemplo simples do funcionamento desse modelo de diagrama.

Figura 11 – Exemplo de um diagrama de tempo



Fonte: Gilleanes, 2011.

3.4 Diagrama de casos de uso e diagrama de atividade

Entre as documentações consideradas para fins de desenvolvimento deste trabalho, foram selecionados dois diagramas não mencionados anteriormente: o diagrama de casos de uso e o diagrama de atividade. Esses diagramas foram escolhidos para proporcionar uma melhor compreensão do funcionamento do software RegNutes.

Embora outros diagramas possam se apresentar bem elaborados e úteis, o diagrama de casos de uso e o diagrama de atividade atendem, de maneira eficaz, ao objetivo pretendido nesse estudo, oferecendo aos programadores uma visão geral do sistema, de suas funcionalidades e do fluxo dos processos internos. Juntos, eles visam proporcionar uma compreensão abrangente do sistema, facilitando a manutenção e a evolução do software, especialmente quando se tornar legado.

Nesse contexto, os tópicos subsequentes realizam uma análise mais aprofundada dos elementos e funcionalidades desses dois tipos de diagramas, detalhando suas figuras e representações para a compreensão do funcionamento do sistema.

3.4.1 Diagrama de casos de uso

O diagrama de casos de uso pode ser definido como aquele que modela o comportamento e ajuda a capturar os requisitos do sistema. De acordo com Larman (2005,

p. 100), “o diagrama de casos de uso é uma ferramenta essencial para identificar e descrever as funcionalidades de um sistema do ponto de vista do usuário, pois ajuda a entender os requisitos e as interações entre os atores e o sistema”.

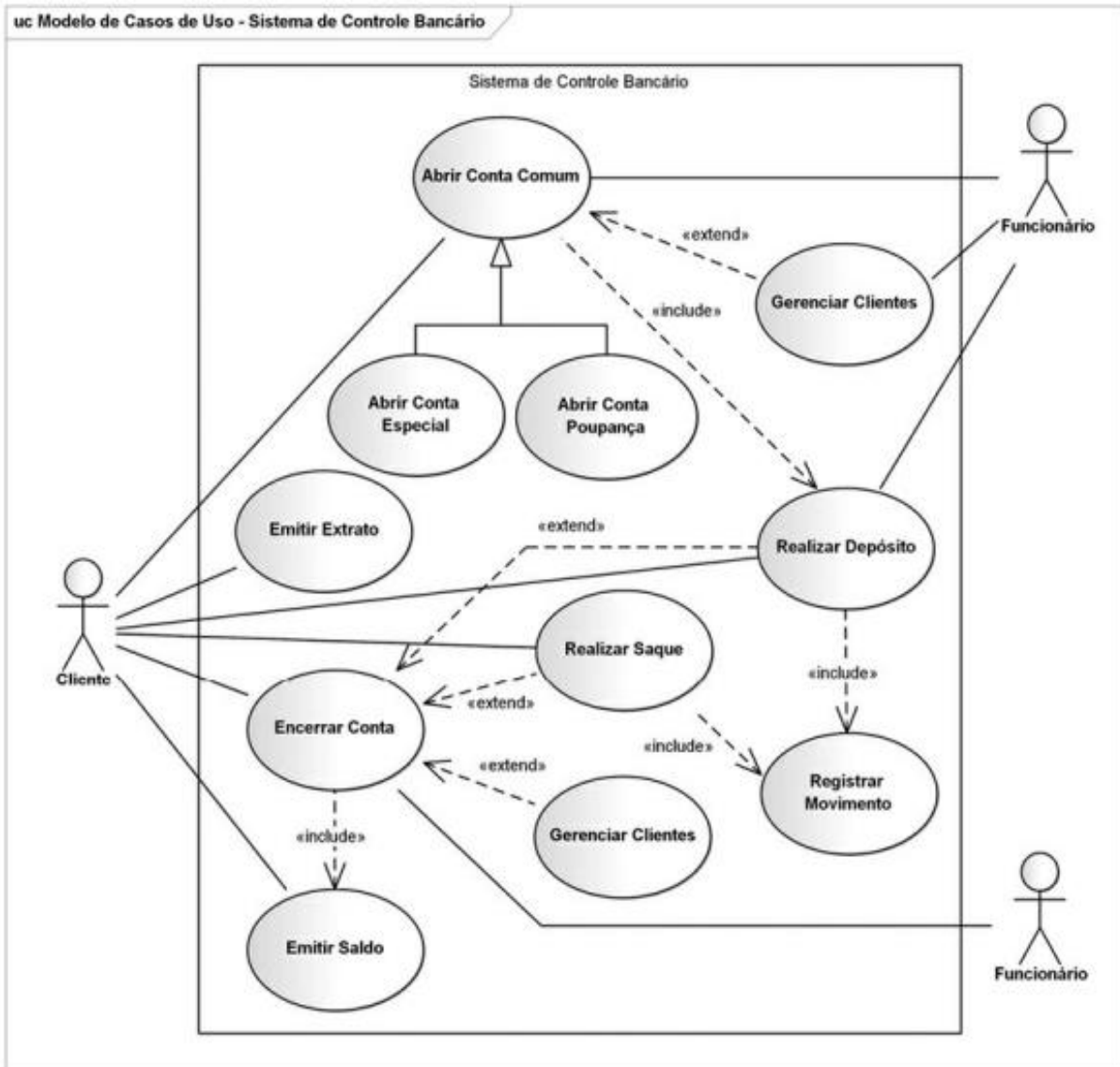
Assim, é possível dizer que esse modelo de diagrama serve para ilustrar e definir o contexto e os requisitos de partes importantes ou do sistema inteiro, identificando as necessidades de cada ator.

Complementando essa ideia, Gilleanes (2011) explica que o diagrama de casos de uso pode ser considerado um tipo de diagrama mais genérico, comumente utilizado em fases de levantamento e análise de requisitos do sistema, mas também relevante para ser consultado durante todo o processo de modelagem, posto que serve de base para outros diagramas, sempre buscando apresentar uma linguagem simples e de fácil compreensão para que a equipe possa ter uma ideia geral de como o sistema irá se comportar.

Desse modo, ele procura identificar os atores (sejam eles os usuários, outros sistemas ou, até mesmo, hardwares especiais) que utilizam, de alguma forma, o software, bem como os serviços, ou seja, as funcionalidades que o sistema disponibilizará aos atores, conhecidas neste diagrama como “casos de uso”.

No entanto, não existe um formato específico de documentação para casos de uso definidos pela UML, o que está de acordo com a característica do próprio diagrama. Isso significa que o formato de documentação de um caso de uso é bastante flexível, permitindo que se documente o caso de uso da forma que se considerar melhor, inclusive mediante uso de pseudocódigo, embora isso fuja bastante do objetivo principal do diagrama, que é utilizar uma linguagem simples para que seja compreendido até por usuários leigos. Um exemplo de construção desse modelo de diagrama pode ser visto na Figura 12.

Figura 12 – Exemplo de um diagrama de casos de uso

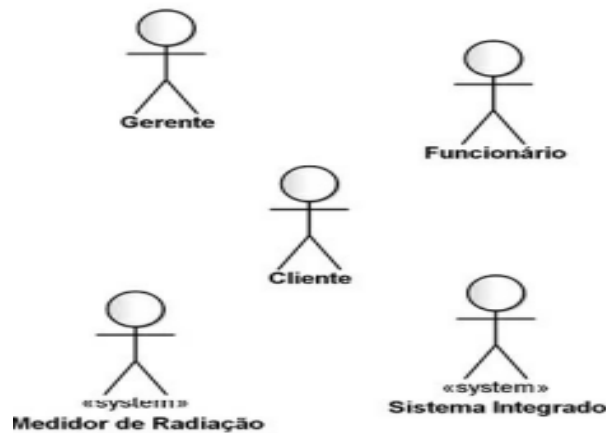


Fonte: Gilleanes, 2011.

3.4.1.1 Atores

No contexto de casos de uso, atores são entidades externas que interagem com o sistema, como usuários, outros sistemas ou dispositivos, e iniciam ou recebem ações do sistema, como pode ser visto na Figura 13.

Figura 13 – Exemplo de atores de um diagrama de casos de uso



Fonte: Gilleanes, 2011.

Booch, Rumbaugh e Jacobson (2005, p. 49) descrevem os atores como “entidades externas ao sistema, mas que se comunicam com ele para realizar alguma funcionalidade”. Assim, é possível dizer que os candidatos a atores devem ser listados, sendo necessário atribuir responsabilidades, objetivos e metas que cada ator busca alcançar ao utilizar o software. Caso não seja possível atribuir um objetivo claro a um ator, é improvável que ele seja um ator válido, devendo ser descartado.

3.4.1.2 Casos de uso

Segundo Larman (2005, p. 108), “os casos de uso descrevem as funcionalidades que o sistema oferece aos atores, representando as interações necessárias para alcançar um objetivo”. São normalmente utilizados para capturar os requisitos do sistema, descrevendo os serviços, tarefas ou funcionalidades que o software deve oferecer. Uma representação simples de casos de uso pode ser visualizada na Figura 14.

Figura 14 – Exemplo de caso de uso



Fonte: Gilleanes, 2011.

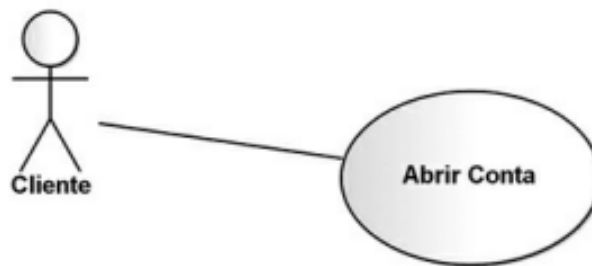
3.4.1.3 Relações

Rumbaugh, Jacobson e Booch (2005, p. 122) definem as funcionalidades das relações da seguinte forma:

As relações entre atores e casos de uso, como associações, generalizações, inclusões e extensões, ajudam a estruturar e organizar a conexão entre funcionalidades. Essas relações ajudam a organizar os casos de uso e entender a hierarquia e a dependência entre funcionalidades.

As associações são aquelas que representam a comunicação direta entre um ator e um caso de uso, demonstrando a interação que ocorre. Essa relação básica é essencial para mapear como um ator participa de um processo no sistema, como se observa na Figura 15.

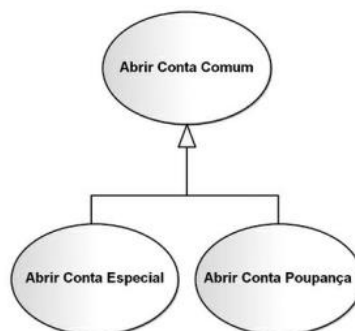
Figura 15 – Exemplo de relação por associação



Fonte: Gilleanes, 2011.

As generalizações, por sua vez, definem uma relação hierárquica entre atores ou casos de uso, na qual um elemento mais específico herda comportamentos de um mais genérico. Isso permite simplificar a modelagem ao evitar duplicação de funcionalidades comuns, conforme Figura 16.

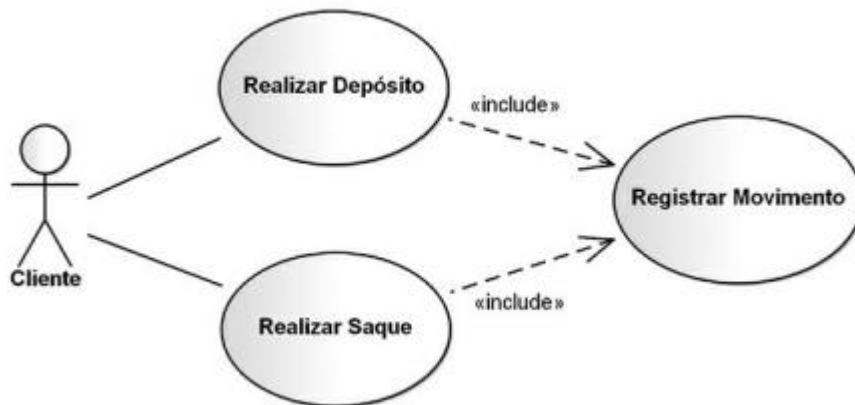
Figura 16 – Exemplo de relação por generalização



Fonte: Gilleanes, 2011.

Já as inclusões representam funcionalidades que sempre fazem parte de outro caso de uso. Esse tipo de relação reflete casos de uso comuns que são incorporados em vários outros, otimizando a reutilização de processos, conforme Figura 17.

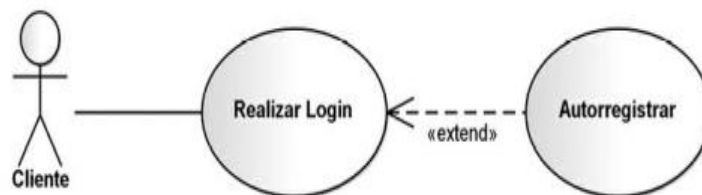
Figura 17 – Exemplo de relação por inclusão



Fonte: Gilleanes, 2011.

Finalmente, as extensões indicam casos de uso opcionais ou alternativos que complementam um caso de uso principal. Essas funcionalidades podem ser executadas dependendo de condições específicas, oferecendo flexibilidade ao sistema, cujo exemplo se observa na Figura 18.

Figura 18 – Exemplo de relação por extensão



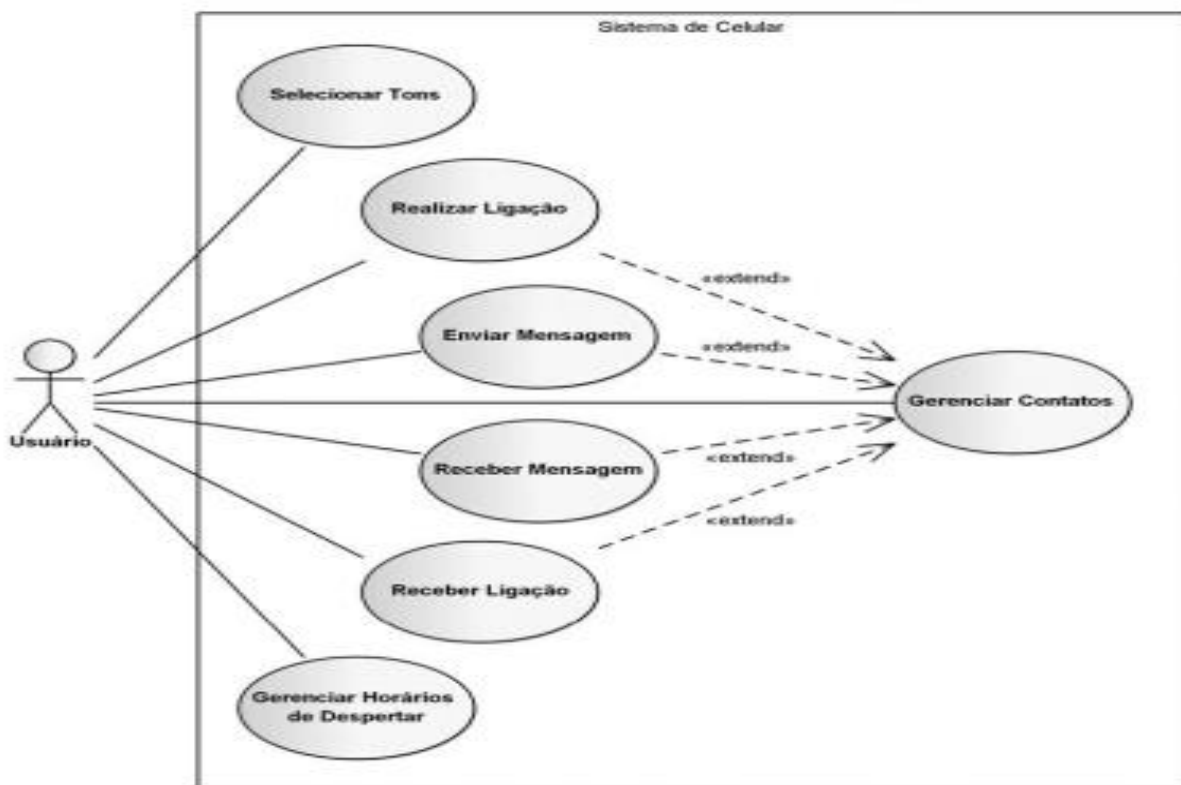
Fonte: Gilleanes, 2011.

Essas relações são essenciais para entender as interações e hierarquias no diagrama de casos de uso.

3.4.1.4 Sistema

O sistema, no diagrama de casos de uso, é representado como uma caixa que delimita a fronteira entre os atores externos e as funcionalidades internas. Como destaca Fowler (2004, p. 69), “o sistema é representado como uma caixa que contém os casos de uso e delimita a fronteira entre os atores e as funcionalidades internas”. Um exemplo de sistema pode ser visto na Figura 19.

Figura 19 – Exemplo de sistema



Fonte: Gilleanes, 2011.

3.4.2 Diagrama de atividade

A partir do diagrama de atividade, vislumbra-se a existência de uma preocupação na descrição de cada passo a ser percorrido para a conclusão de uma atividade específica, podendo esta ser representada por um método com certo grau de complexidade, um algoritmo ou um processo completo.

Segundo Rumbaugh, Jacobson e Booch (2005, p. 45):

É possível dizer que o diagrama de atividade tende-se a concentrar-se na representação do fluxo de controle de uma atividade, assim buscando de forma

mais detalhada definir e representar como as coisas funcionam de forma clara e simples e como elas se relacionam dentro do processo.

A modelagem de atividade destaca a sequência, ou seja, a ordem em que as atividades devem ocorrer, e condições, que são regras que determinam quando e como essas atividades devem ser executadas, para coordenar e organizar ações mais detalhadas dentro do sistema.

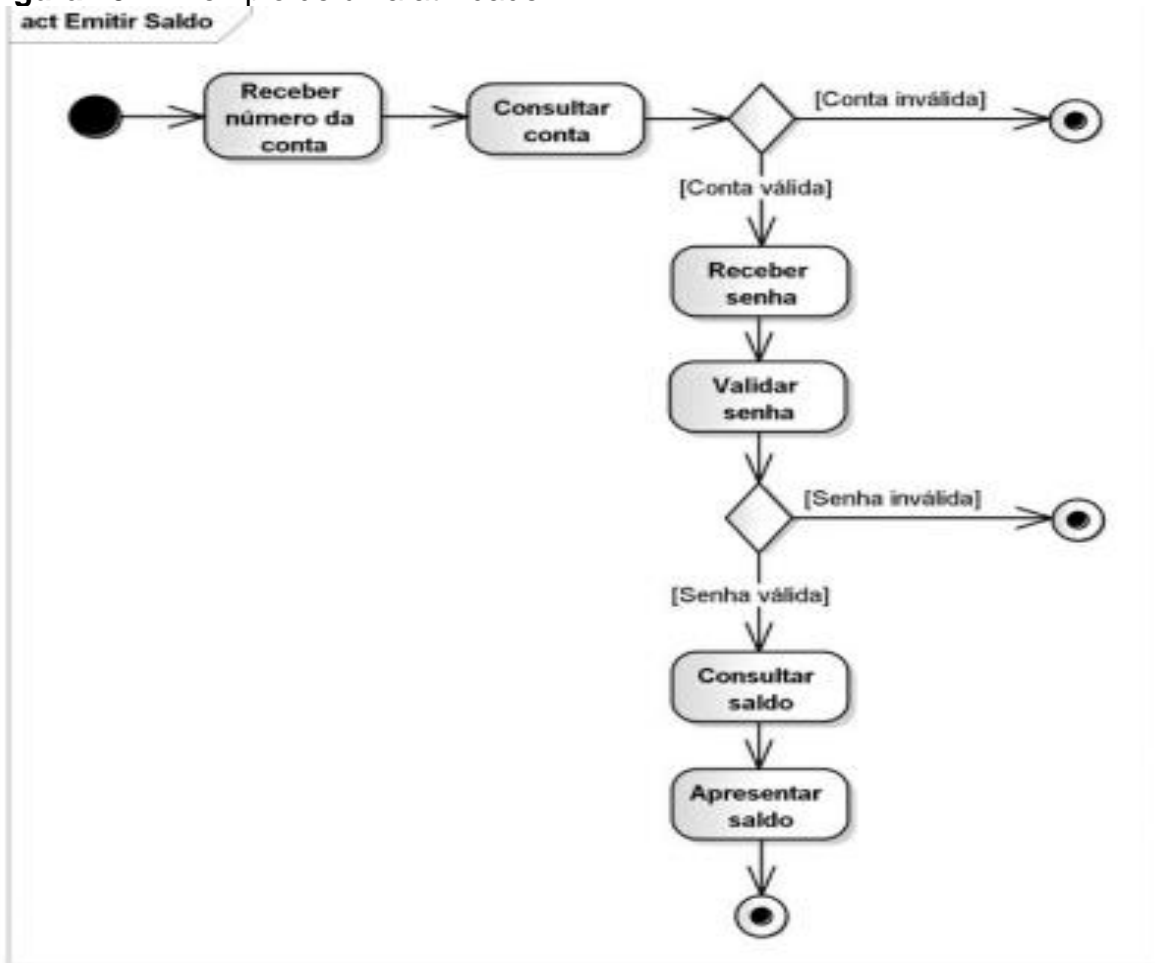
Dessa forma, conforme explica Gilleanes (2011, p. 391), “o diagrama de atividade é o diagrama com maior ênfase ao nível de algoritmo da UML, e provavelmente um dos mais detalhistas”. Esse diagrama apresenta semelhanças com os antigos fluxogramas utilizados para desenvolver a lógica de programação e determinar o fluxo de controle de um algoritmo.

Assim, o diagrama de atividade é utilizado para modelar atividades que podem ser um método, um algoritmo ou um processo completo. Nesse contexto, elas são os métodos correspondentes às aplicadas à modelagem organizacional para engenharia de processos de negócios e *workflow*.

Ressalte-se que atividades podem ser também usadas para modelagem de sistemas de informação com o fim de especificar processos ao nível de sistema. Logo, o diagrama pode ser representado de forma completa ou de forma mais parcial para detalhar mais de uma metodologia aplicada dentro do fluxo.

Com base nessas noções, é necessário detalhar o uso desses elementos do diagrama para que possa ser utilizado de forma correta, bem como padroniza-lo para que não seja aplicado de forma errônea. Um exemplo de diagrama de atividade pode ser visualizado na Figura 20:

Figura 20 – Exemplo de uma atividade



Fonte: Gilleanes, 2011.

3.4.2.1 Atividades

No sistema, as atividades representam as ações ou tarefas realizadas, detalhando o seu comportamento ao longo do tempo. Como Booch, Rumbaugh e Jacobson (2005, p. 139) explicam, “as atividades modelam o comportamento do sistema, detalhando como as ações são executadas e como o fluxo de trabalho é controlado”.

Esses comportamentos podem ser desencadeados pela conclusão de outras atividades, pela disponibilidade de objetos e dados ou pela ocorrência de eventos externos, os quais influenciam no fluxo de execução do sistema. Um exemplo de sua forma pode ser encontrado na Figura 21.

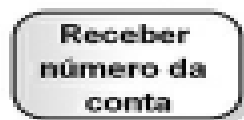
Figura 21 – Exemplo de uma atividade

Fonte: Gilleanes, 2011.

3.4.2.2 Nó de ação

O nó de ação, em um diagrama de atividades, representa uma unidade de trabalho ou operação que é executada dentro do fluxo de um processo. Segundo Rumbaugh, Jacobson e Booch (2005, p. 2017), “um nó de ação descreve uma tarefa indivisível e atômica dentro do contexto de uma atividade, definindo um ponto específico onde o sistema executa um comportamento”.

Essas ações são fundamentais para capturar o comportamento detalhado de um sistema, pois indicam precisamente onde e como as operações ocorrem, garantindo que o fluxo de atividades seja claramente delineado e compreensível no contexto geral do processo modelado, cujo exemplo pode ser visto na Figura 22.

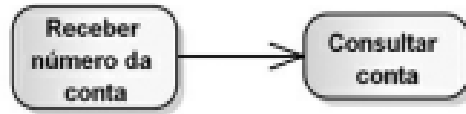
Figura 22 – Exemplo de um nó de ação

Fonte: Gilleanes, 2011.

3.4.2.3 Fluxo de controle

O fluxo de controle em um diagrama de atividades é responsável por determinar a sequência na qual as ações e atividades são executadas. Em si, o fluxo de controle busca conectar as ações e atividades, determinando assim, a ordem e o caminho pelo qual o processo se desenvolve, assegurando que as atividades ocorram na sequência devida (UML Distilled, 2004, p. 106). Ele é representado por setas que indicam o direcionamento das operações dentro do sistema, assegurando que o comportamento desejado seja seguido ao longo do processo modelado. O funcionamento do nó de ação pode ser encontrado na figura 23.

Figura 23 – Exemplo de um fluxo de controle

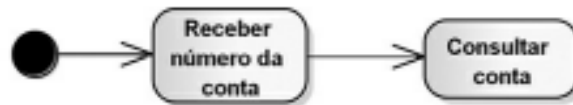


Fonte: Gilleanes, 2011.

3.4.2.4 Nó inicial

O nó inicial, representado por um círculo sólido, marca o ponto de partida de uma atividade ou processo dentro do diagrama. Segundo Larman (2005, p. 138), “o nó inicial indica onde o fluxo de atividades começa, servindo como o ponto de entrada para o comportamento do sistema”. Este nó não possui predecessores e é o primeiro a ser ativado, desencadeando as ações subsequentes no modelo. Uma abordagem prática pode ser vista na Figura 24.

Figura 24 – Exemplo de uma abordagem com nó inicial



Fonte: Gilleanes, 2011.

3.4.2.5 Nó de final de atividade

O nó de final de atividade, representado por um círculo com outro círculo concêntrico, indica o término de todas as atividades associadas em um processo. Conforme Rumbaugh, Jacobson e Booch (2005, p. 142), “o nó de final de atividade encerra todos os fluxos que estão em execução dentro de uma atividade, sendo considerada a última interação que aquele fluxo ainda tinha”. Esse nó assegura que o sistema conclua corretamente todas as operações antes de finalizar e seu exemplo pode ser visto na Figura 25.

Figura 25 – Exemplo de um nó de final de atividade

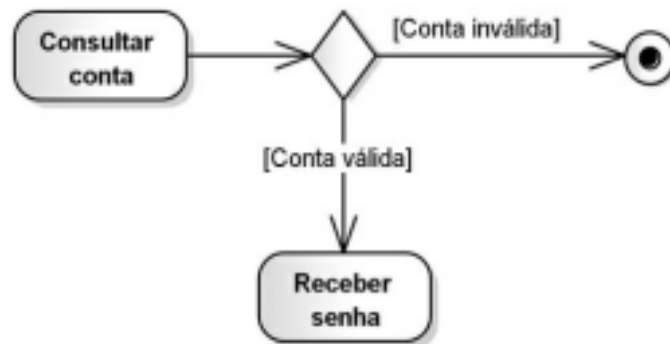


Fonte: Gilleanes, 2011.

3.4.2.6 Nó de Decisão

O nó de decisão, representado por um losango, é utilizado para modelar pontos no processo em que diferentes caminhos de execução são possíveis. Segundo Rumbaugh, Jacobson e Booch (2005, p. 229), “o nó de decisão define bifurcações lógicas no fluxo de controle, onde o comportamento subsequente depende de condições específicas”. Isso permite que o modelo capture cenários em que a execução varia com base em entradas ou estados do sistema. Exemplo de um nó pode ser analisado na Figura 26.

Figura 26 – Exemplo de um nó de decisão

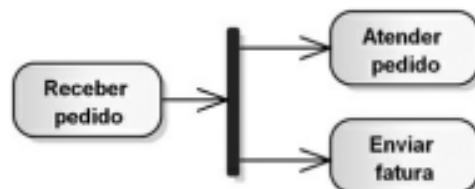


Fonte: Gilleanes, 2011.

3.4.2.7 Nó de Bifurcação e de União

O nó de bifurcação, também representado por uma linha preta com setas paralelas, pode ser observado na Figura 27, permitindo que o fluxo de controle seja dividido em vários caminhos paralelos.

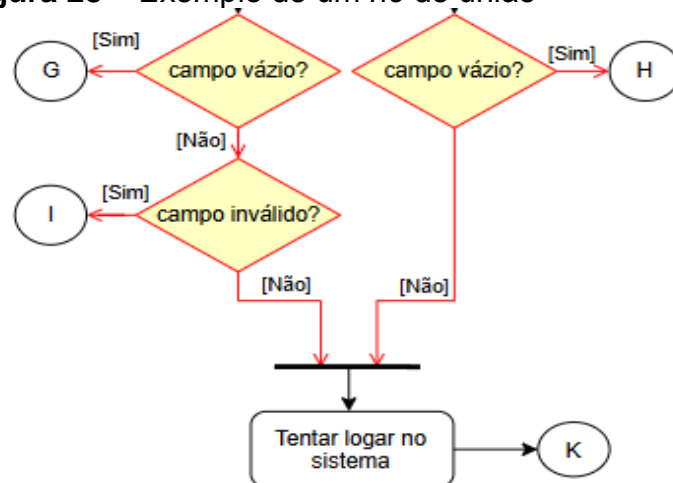
Figura 27 – Exemplo de um nó de bifurcação



Fonte: Gilleanes, 2011.

A mesma representação ocorre no nó de união. Porém, como demonstrado na Figura 28, as setas sintetizam para formar uma nova.

Figura 28 – Exemplo de um nó de união



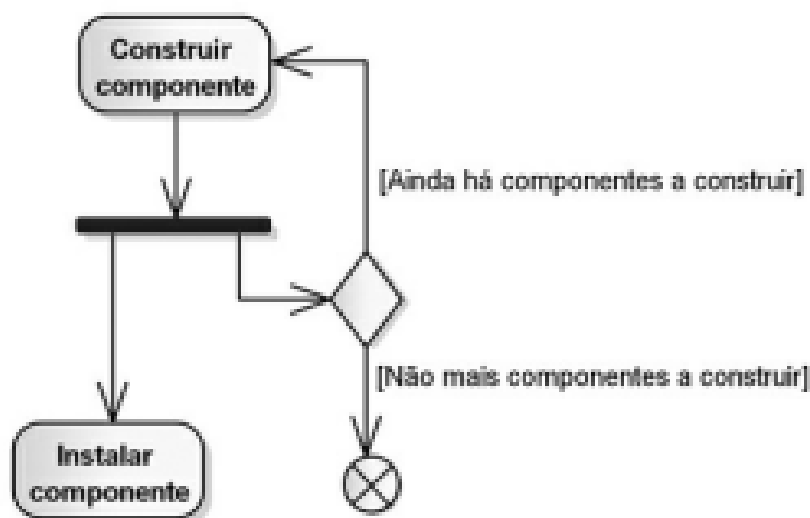
Fonte: Elaboração do Autor, 2024.

Conforme Rumbaugh, Jacobson e Booch (2005, p. 144), “o caso do nó de bifurcação se deve a criação de fluxos de controle paralelos, permitindo assim que múltiplas atividades ocorram simultaneamente”.

3.4.2.8 Nó de final de fluxo

O nó de final de fluxo, representado por um círculo com um X dentro, conforme exemplificado na Figura 29, encerra um fluxo específico sem finalizar a atividade completa.

Figura 29 – Exemplo de um nó de final de fluxo



Fonte: Gilleanes, 2011.

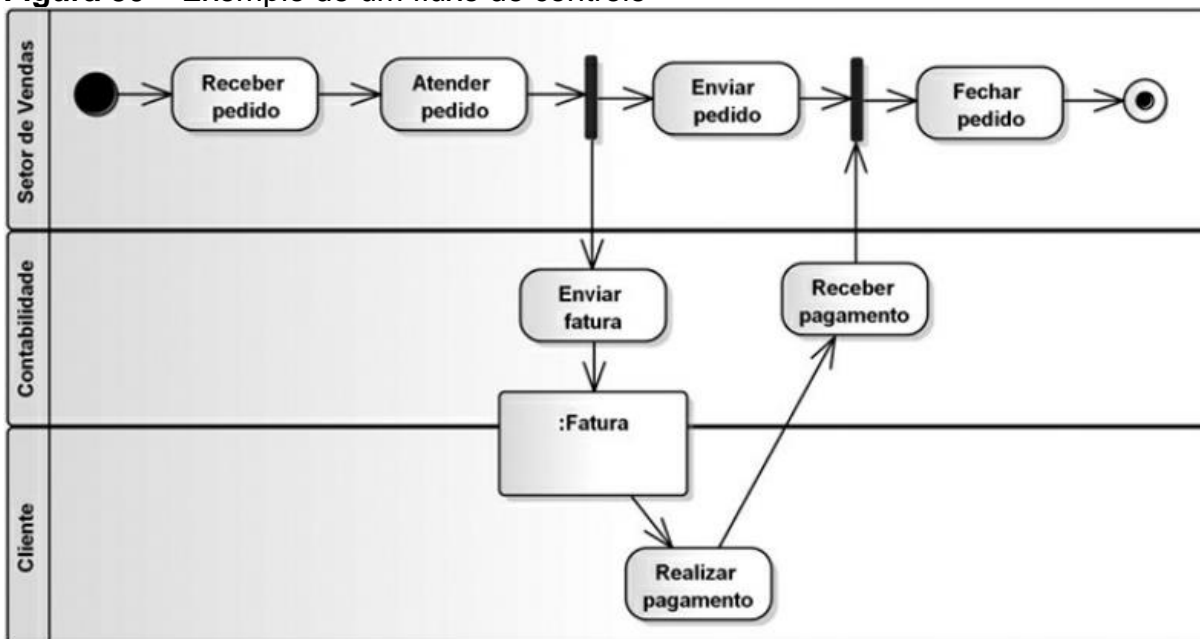
Conforme Fowler (2004, p. 110), “o nó de final de fluxo permite que partes do processo sejam concluídas enquanto outras continuam, encerrando apenas os fluxos individuais”. Isso proporciona flexibilidade ao modelo, permitindo o término parcial de processos sem interferir na atividade como um todo.

3.4.2.9 Fluxo de controle

As partições de atividade, também conhecidas como *swimlanes*, organizam as atividades de acordo com os responsáveis por sua execução, dividindo o diagrama em áreas que indicam quem ou o que é responsável por cada ação ou conjunto de ações.

Essa organização facilita a visualização das responsabilidades dentro do processo e torna mais clara a interação entre diferentes agentes ou componentes, contribuindo para uma melhor compreensão do fluxo de trabalho. Seu exemplo pode ser visualizado na Figura 30.

Figura 30 – Exemplo de um fluxo de controle



Fonte: Gilleanes, 2011.

3.4.2.10 Conectores

Os conectores são elementos que possibilitam a continuidade do fluxo de controle em diferentes partes do diagrama, sendo especialmente úteis em diagramas complexos. Eles simplificam o layout ao permitir a ligação de fluxos de controle que não estão diretamente conectados, evitando o cruzamento de linhas e, assim, preservando a clareza e a legibilidade do diagrama.

Esses elementos são fundamentais para manter a coesão e a facilidade de interpretação em diagramas mais extensos e seu uso pode ser visto na Figura 31.

Figura 31 – Exemplo de um conector



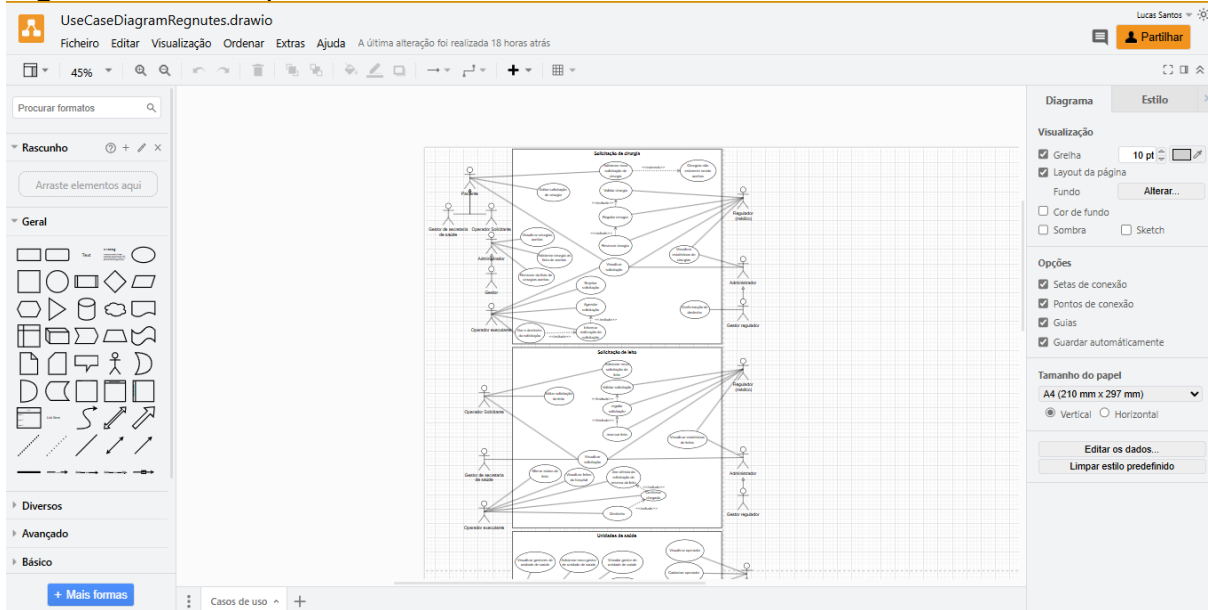
Fonte: Gilleanes, 2011.

3.5 Draw.io

O Draw.io é uma ferramenta online que permite a criação de diversos tipos de diagramas, como diagramas de atividades e de casos de uso, sendo muito utilizado por equipes de desenvolvimento para documentar e planejar sistemas de forma visual e colaborativa (Seibert, 2023). Destaca-se pela sua disponibilidade, permitindo o acesso direto via navegador sem necessidade de instalação de software. Também suporta o armazenamento de arquivos na nuvem e, em alguns casos, o trabalho offline.

A ferramenta oferece colaboração em tempo real, facilitando o trabalho em equipe, especialmente para equipes remotas. Com uma ampla biblioteca de formas e opções de personalização, o draw.io atende às necessidades específicas de usuários, como *technical writers* e engenheiros de software. Essa ferramenta pode ser observada na Figura 32.

Figura 32 – Exemplo da ferramenta Draw.io



Fonte: Elaborado pelo autor, 2024.

3.6 RegNUTES

Para que este trabalho possa ser discutido de forma apropriada, faz-se necessário, nesse ponto, contextualizar a fonte principal de sua orientação: O RegNUTES.

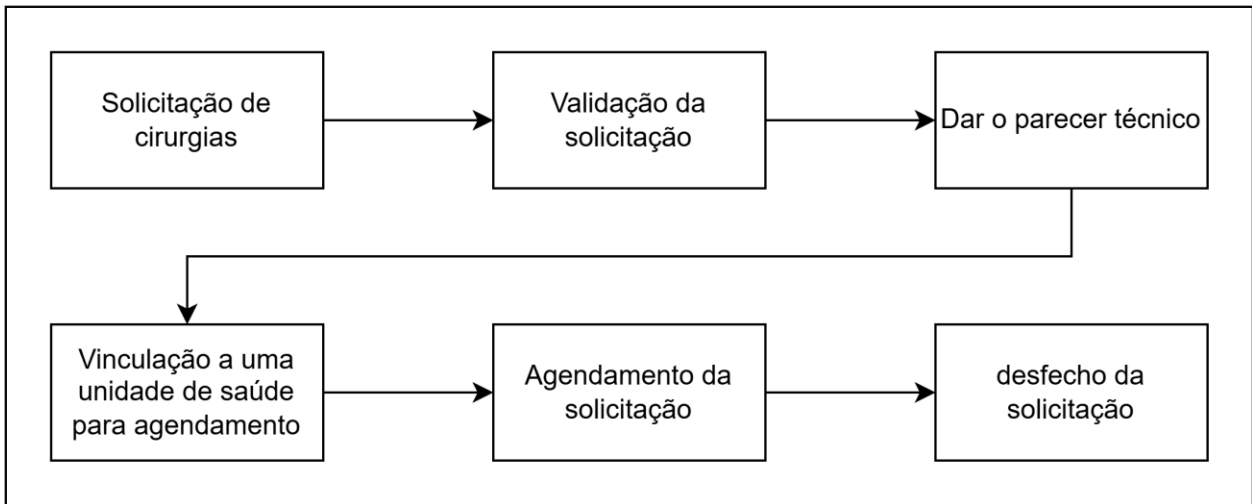
Em síntese, o RegNUTES é uma plataforma de regulação que faz parte do Sistema de Regulação da Secretaria de Saúde do Estado da Paraíba. Ela é utilizada, principalmente, para organizar e gerenciar a fila de cirurgias eletivas no estado, garantindo que os pacientes sejam atendidos de maneira eficiente e dentro dos prazos estipulados.

O sistema permite o monitoramento e a organização dos processos, desde a entrada do paciente na fila de espera até a realização da cirurgia, buscando reduzir filas e otimizar o atendimento. Ressalte-se que o RegNUTES tem, em resumo, duas funcionalidades principais: cirurgias e leitos.

3.6.1 Cirurgias

O fluxo de cirurgias busca, de forma simples e prática, fazer o processo de cadastro, seguindo até o desfecho. Ele se prende a poucas informações no momento da criação da solicitação, sendo as mais importantes: o paciente que foi cadastrado no sistema, a cirurgia que vai ser realizada e os exames que foram feitos. A partir desses dados, o médico executante pode analisar e definir o processo está seguindo de forma correta. É possível ver uma demonstração do funcionamento do fluxo na figura 33.

Figura 33 – Exemplo do fluxo de cirurgias



Fonte: Elaborado pelo autor, 2024.

3.6.1.1 Cadastro da solicitação

Essa etapa perpassa o cadastro de informações necessárias pelo operador solicitante, mormente o gestor da unidade de saúde. Tais informações são cadastradas diretamente em um formulário disponibilizado na listagem da Figura 34.

Figura 34 – Exemplo de cadastro de uma solicitação

A imagem mostra uma interface de usuário de um sistema web. No topo, há um cabeçalho com o logo 'regNUTES' e o título 'SOLICITAÇÃO DE CIRURGIA'. Abaixo, há um formulário com os seguintes campos:

- Paciente:** João Vitor Rocha Vicenca
- Dados do paciente:**
 - Nome: João Vitor Rocha Vicenca
 - CPF: 600.319.840-02
 - Cartão do SUS: -
 - Email: -
 - Telefone: (33) 33333-3333
 - Data de nascimento: 01/03/2000
 - CEP: 58416080
 - Rua: Rua Maria do Socorro Brandão
 - Número: 33
 - Bairro: Jardim Quarenta
 - Cidade: Campina Grande
 - Estado: PB
 - Complemento: -
- Cirurgia:** RETIRADA DE CORPO ESTRANHO DA COLUNA CERVICAL POR VIA ANTERIOR
- Dados da cirurgia:**
 - Nome: RETIRADA DE CORPO ESTRANHO DA COLUNA CERVICAL POR VIA ANTERIOR
 - Código SUS: 04.08.03.057-7
- Documentos(s):** Um ícone de documento com o nome 'whatsapp image 2024-03-20 às 13:25:12_302380ee.jpg' e um botão '+ Novo documento'.

Na base do formulário, há botões 'CANCELAR' e 'SALVAR'. À direita, há uma barra lateral com opções como 'ALTERAR DATA', 'ADICIONAR +', e 'Ações/Alertas'.

Fonte: Elaborado pelo autor, 2024.

Importante ressaltar que o paciente somente pode cadastrar, por si mesmo, uma solicitação a partir do sistema chamado Opera PB.

3.6.1.2 Validação e parecer médico

Após o cadastro, o regulador médico poderá realizar a validação, o que pode ser feito quando o formulário for solicitado, como na Figura 35.

Figura 35 – Exemplo de formulário com opção de validar solicitação

The screenshot shows a web interface for a surgical request form. The form is titled 'SOLICITAÇÃO DE CIRURGIA' and is displayed in a modal window. The background shows a sidebar with 'SOLICITAÇÕES' and a search filter. The form content includes:

- Information: Solicitação cadastrada em 20/08/2024 às 15:57.
- UNIDADE DE ORIGEM: CLIPSI Hospital Geral de Campina Grande, Campina Grande, PB, Rua Treze de Maio.
- Registro: 20240820155730430 | Estado solicitação: Solicitada
- Dados do paciente:

Nome	CPF	Cartão do SUS	
Joao Vitor Rocha Vicenca	600.319.840-02		
Email	Telefone	Data de nascimento	
-	(33) 33333-3333	01/03/2000	
CEP	Rua	Número	Bairro
58416080	Rua Maria do Socorro Brandão	33	Jardim Quarenta
Cidade	Estado	Complemento	
Campina Grande	PB	-	
- Parecer médico: dropdown menu.
- Documentos(s): Nenhum documento adicionado.
- Buttons: CHAT, VALIDAR.

Fonte: Elaborado pelo autor, 2024.

Após essa fase, a solicitação pede o fornecimento de um parecer, podendo ser preenchido como na Figura 36. Ressalte-se que apenas o regulador, ou seja, um médico, pode realizar o parecer.

Figura 36 – Exemplo de formulário para informar o parecer técnico

regNUTES

Nome: Joao Vitor Rocha Vicenca | CPF: 600.319.840-02 | Cartão do SUS: -

Email: - | Telefone: (33) 33333-3333 | Data de nascimento: 01/03/2000

CEP: 58416080 | Rua: Rua Maria do Socorro Brandão | Número: 33 | Bairro: Jardim Quarenta

Cidade: Campina Grande | Estado: PB | Complemento: -

Parecer médico

Paciente não preenche critérios para regulação

Prioridade*

Atual Verde Amarelo Laranja Vermelho

Grupo*

REVISÃO DE ARTROSE / TRATAMENTO CIRÚRGICO DE PSEUDARTROSE DA COLLINA CERVICAL POSTERIOR

Justificativa*

Paciente deve ser acompanhado por meio medicamentoso

Documentos(s)

CANCELAR SALVAR ✓

Fonte: Elaborado pelo autor, 2024.

3.6.1.3 Reserva da solicitação

Quando a solicitação é corretamente regulada, o regulador pode reservar uma vaga em uma Unidade de Saúde para o agendamento da cirurgia. Porém, para que a Unidade apareça como opção de reserva, ela precisa ter vagas disponíveis, as quais são cadastradas pelo gestor hospitalar, como se visualiza na Figura 37. Isso adiciona uma camada extra de requisitos ao sistema.

Figura 37 – Exemplo de formulário para reservar a cirurgia

regNUTES

Solicitação cadastrada em 09/11/2023 às 12:40

UNIDADE DE ORIGEM
Secretaria Municipal de Saúde de Campina Grande

Registro: 2023110812404314 | Estado solicitação: Em andamento

Dados do paciente

Nome: - | CPF: - | Cartão do SUS: -

RESERVAR CIRURGIA

Clique na cirurgia da unidade de saúde que deseja reservar, para finalizar confirme a reserva.

CLIPSI Hospital Geral de Campina Grande

EXTIRPAÇÃO DE BOCIO INTRATORACICO POR VIA TRANSESTERNAL

Documentos(s)

Nenhum documento adicionado

CHAT RESERVAR CIRURGIA EDITAR

07/11/2023 15:21:01 de Paula Campina Grande CIRURGICO DE PSEUDARTROSE DA COLLINA TORACO LOMBO SACRA ANTERIOR Em andamento

Fonte: Elaborado pelo autor, 2024.

3.6.1.4 Agendamento da cirurgia

Com a conclusão do processo anterior, a solicitação chega à Unidade de Saúde, onde é realizado o agendamento. Nessa etapa, são informados a data, o médico responsável pela cirurgia e eventuais observações adicionais, cujo exemplo pode ser visto na Figura 38.

Figura 38 – Exemplo de agendamento de cirurgia

The screenshot displays the regNUTES web application interface. A modal window titled 'CIRURGIA' is open, showing the following details:

- Nome:** EXTIRPAÇÃO DE BOCIO INTRATORACICO POR VIA TRANSESTERNAL (04.02.01.001-9)
- Status da cirurgia:** Pendente
- Nome do médico*:** Dr. Carlos Fortunado
- Data de agendamento*:** 31/08/2024
- Observações:** (Empty text area)

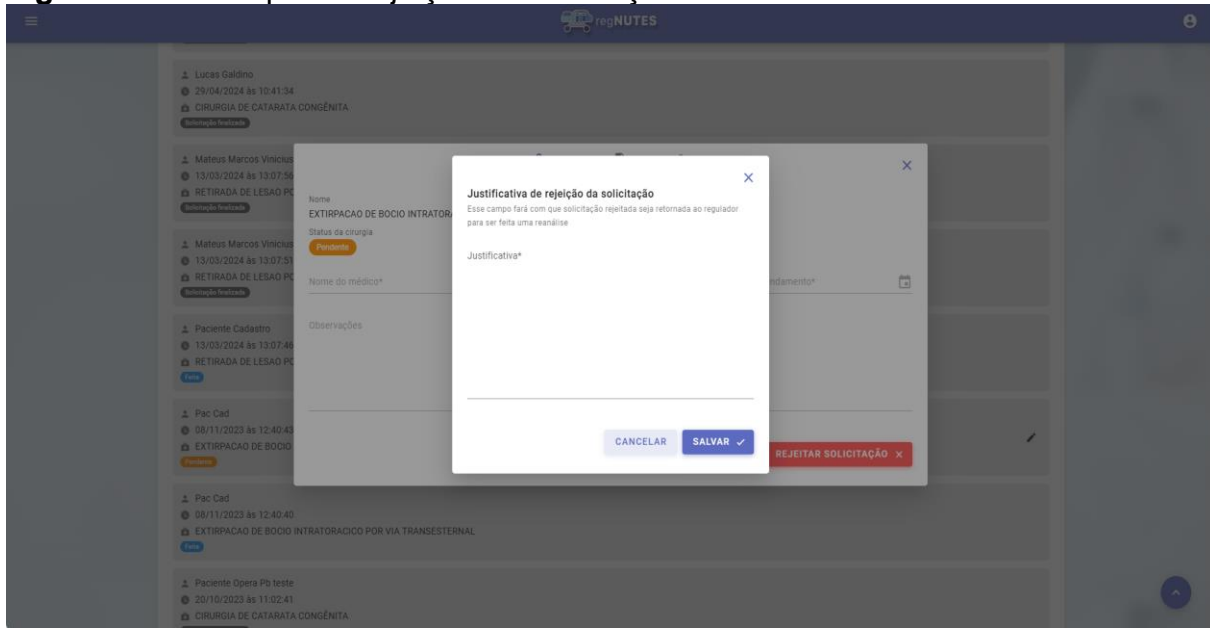
At the bottom of the modal, there are two buttons: 'CANCELAR' and 'SALVAR ✓'. The background shows a list of medical records with columns for patient name, date, and procedure name.

Fonte: Elaborado pelo autor, 2024.

3.6.1.5 Rejeição

Um fluxo alternativo que pode acontecer após a chegada na Unidade, a solicitação pode ser rejeitada por motivos internos, os quais fazem a solicitação ser analisada novamente pelo regulador. Um exemplo dessa fase pode ser observado na Figura 39.

Figura 39 – Exemplo de rejeição de solicitação

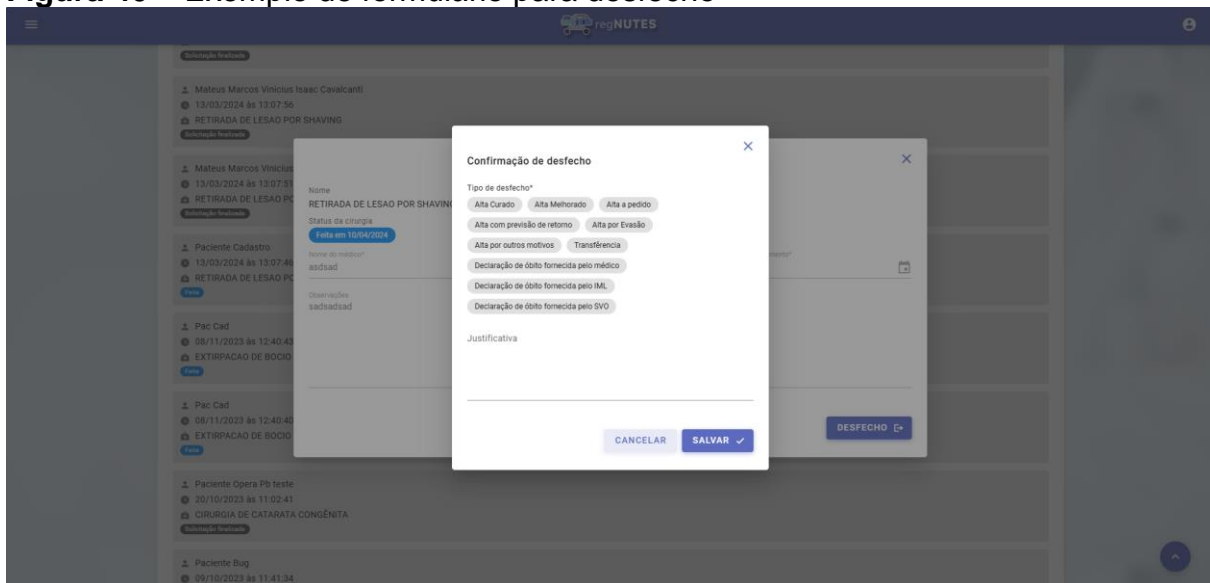


Fonte: Elaborado pelo autor, 2024.

3.6.1.6 Desfecho da solicitação

Se a cirurgia for realizada, o desfecho é registrado, incluindo todas as informações finais sobre o estado do paciente pós-cirúrgico. A solicitação não será apagada do sistema e pode ser revisada em uma guia especial que contém todas as informações, permitindo que o médico avalie o caso em um possível retorno do paciente, como visto na Figura 40.

Figura 40 – Exemplo de formulário para desfecho



Fonte: Elaborado pelo autor, 2024.

3.6.1.7 Finalizar solicitação

A finalização de uma solicitação pode ser definida como uma sumarização, recebendo apenas as informações da Unidade de Saúde que foi executada, a data e o desfecho do caso. Porém, esta etapa somente pode ser realizada pelo gestor regulador.

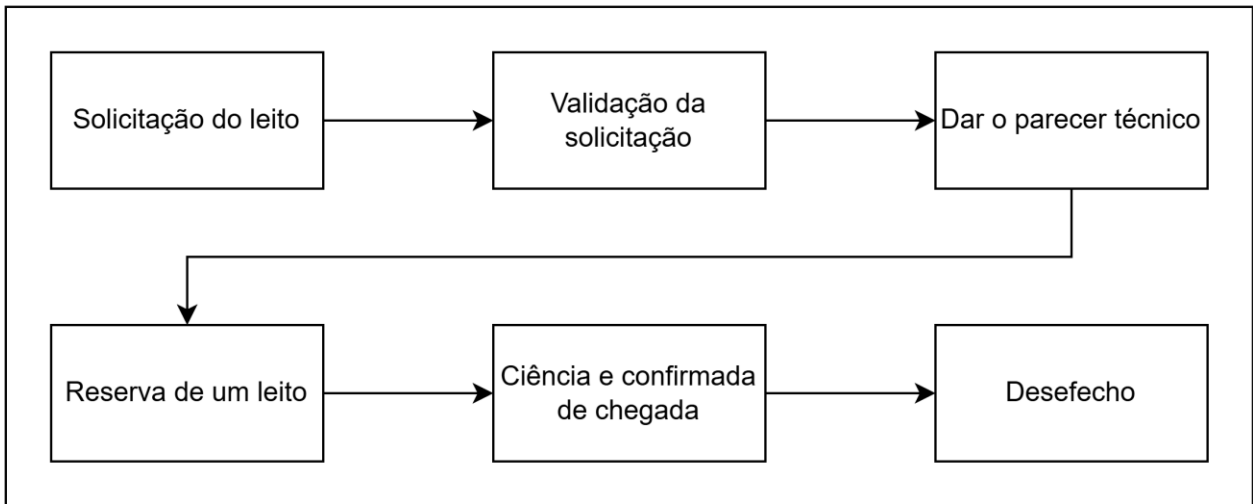
Importante salientar que não há necessidade de que qualquer número de cirurgias esteja vinculado à Unidade de Saúde. A janela de finalização da solicitação pode ser vista na Figura 41.

Figura 41 – Exemplo de formulário para finalizar solicitação

Fonte: Elaborado pelo autor, 2024.

3.6.2 Leitos

Em relação ao fluxo de leitos, existe a necessidade de maiores informações sobre os pacientes para que possam ser tratados de forma efetiva, dentre as quais os sinais vitais, existência de eventual síndrome gripal, comorbidades, oximetria, laboratório, etc. São essas informações que diferenciarão o tratamento efetivo e o inefetivo ao paciente. Sendo possível ver o fluxo na figura 42.

Figura 42 – Exemplo do fluxo de leitos

Fonte: Elaborado pelo autor, 2024.

3.6.2.1 Cadastro da solicitação

O fluxo de leitos também necessita do cadastro de informações do paciente pelo operador solicitante e pelo gestor da Unidade de Saúde, porém, nesse caso, somente essas pessoas podem solicitar.

Todas as informações necessárias a essa etapa podem ser visualizadas na Figura 43, as quais farão parte do prontuário do paciente que chegará ao leito. Note-se que, em cada módulo, há um conjunto de informações que podem ser preenchidas por aqueles que cadastram as solicitações.

Figura 43 – Exemplo de formulário para cadastrar solicitação de leito

SOLICITAÇÃO DE LEITO

Dados do paciente

Paciente: João Vitor Rocha Vicenca

Coração parabeano

Nome	CPF	Cartão do SUS
João Vitor Rocha Vicenca	600.319.840-02	
Email	Telefone	Data de nascimento
-	(33) 3333-3333	01/03/2000
CEP	Rua	Número
58416080	Rua Maria do Socorro Brandão	33
Cidade	Estado	Bairro
Campina Grande	PB	Jardim Quarenta
		Complemento
		-

REMOVER PACIENTE

Sinais vitais e tipo de leito

Síndrome gripal

Comorbidades

Oximetria periférica – Sistema Respiratório e Ventilação

Verificação de PA/FC/FR – Cardiovascular e Infecção

CANCELAR SALVAR ✓

Fonte: Elaborado pelo autor, 2024.

3.6.2.2 Validação e parecer médico

Após o cadastro, o médico pode validar a solicitação a partir do botão “Validar”, conforme se observa na Figura 44, e, então, será possível realizar o cadastro do parecer, assim como ocorre no processo de fluxo de cirurgia, com a informação extra do tipo de leito em que o paciente será colocado, o que pode ser visto na Figura 45. Frise-se que, nesta etapa, somente o regulador, que é um médico, pode realizar a validação.

Figura 44 – Exemplo de formulário para validar solicitação de leito

The screenshot shows a web application interface for 'regNUTES'. A modal window titled 'SOLICITAÇÃO DE LEITO' is open, displaying the following information:

- UNIDADE DE ORIGEM:** CLIPSI Hospital Geral de Campina Grande, Campina Grande, PB, Rua Treze de Maio.
- Registro:** 20230926163513157, Estado solicitação: Solicitada.
- Dados do paciente:**
 - Nome: Paciente Teste, CPF: 266.062.260-40, Cartão do SUS: [blank]
 - Email: -, Telefone: (22) 22222-2777, Data de nascimento: 22/02/1950
 - CEP: 58410000, Rua: Avenida Professor Almeida Barreto, Número: 22, Bairro: Estação Velha
 - Cidade: Campina Grande, Estado: PB, Complemento: -
- Sinais vitais e tipo de leito:**
 - Sinais vitais: Frequência respiratória, Saturação (%), Ar/D^o, Frequência cardíaca, Pressão arterial, Temperatura (C°), Consciência.
 - Tipo de leito: [blank]
- Buttons:** CHAT, VALIDAR.

The background shows a list of records with columns for Registro, Paciente, Cidade, Data de nascimento, and Estado.

Fonte: Elaborado pelo autor, 2024.

Figura 45 – Exemplo de formulário para cadastrar o parecer técnico

The screenshot shows the 'Parecer médico' form in the regNUTES system. The form includes the following fields and options:

- Paciente não preenche critérios para regulação:** [checkbox]
- Prioridade*:** Azul, Verde (selected), Amarela, Laranja, Vermelha.
- Tipo de leito*:** Leito clínico (selected), Leito cirúrgico, Leito de UTI.
- Justificativa*:** Leito para atendimento.
- Anexo(s):** Nenhum documento adicionado.
- Buttons:** CANCELAR, SALVAR.

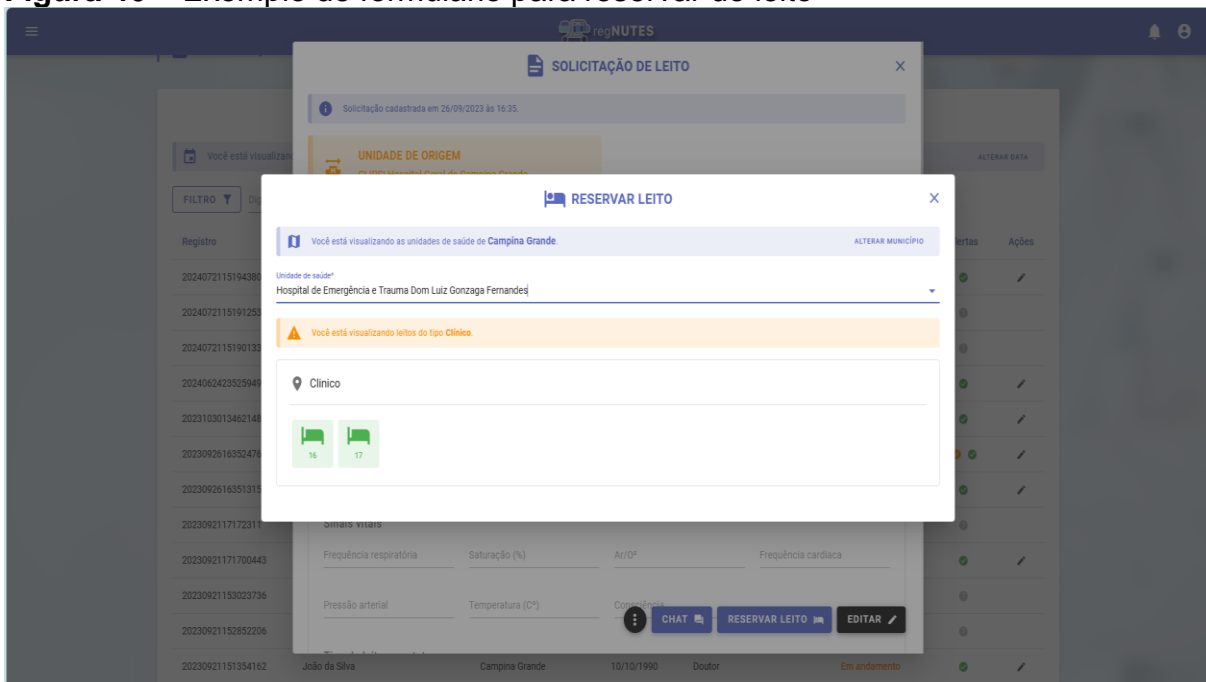
The background shows the same list of records as in Figure 44.

Fonte: Elaborado pelo autor, 2024.

3.6.2.3 Reserva da Solicitação

Quando a solicitação é corretamente regulada, o regulador, após definir o município em que o paciente será internado, pode solicitar a reserva de um leito em uma Unidade de Saúde. Porém, para que a Unidade apareça como opção de reserva, precisa haver alguma vaga na ala do tipo de leito da solicitação, como visto na Figura 46.

Figura 46 – Exemplo de formulário para reservar do leito



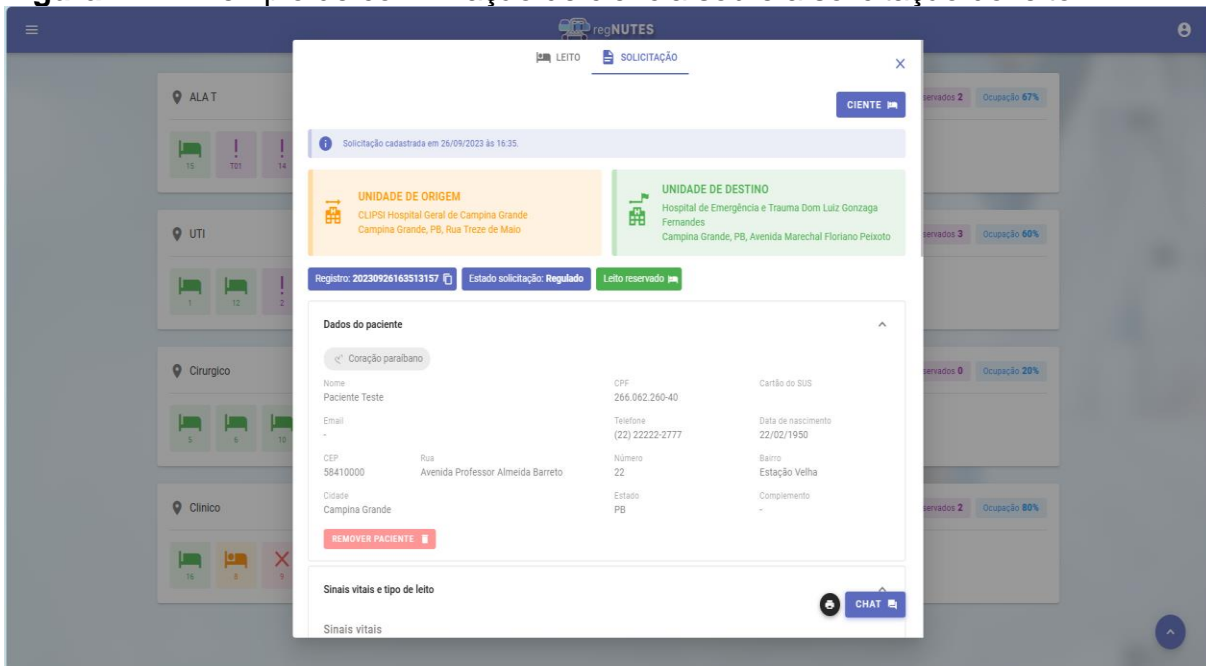
Fonte: Elaborado pelo autor, 2024.

Os dados acerca de leitos vagos são fornecidos pelo operador executante, a partir da liberação de um leito que estava ocupado, ou pelo gestor de Unidade de Saúde, com a criação de uma nova ala e, conseqüentemente, de novos leitos ou com a disponibilização de novo leito em uma ala já existente.

3.6.2.4 Ciência e confirmação a chegada

Quando a reserva do leito é solicitada, o operador executante deve confirmar a ciência da reserva, o que ocorre a partir do botão visualizado na Figura 47.

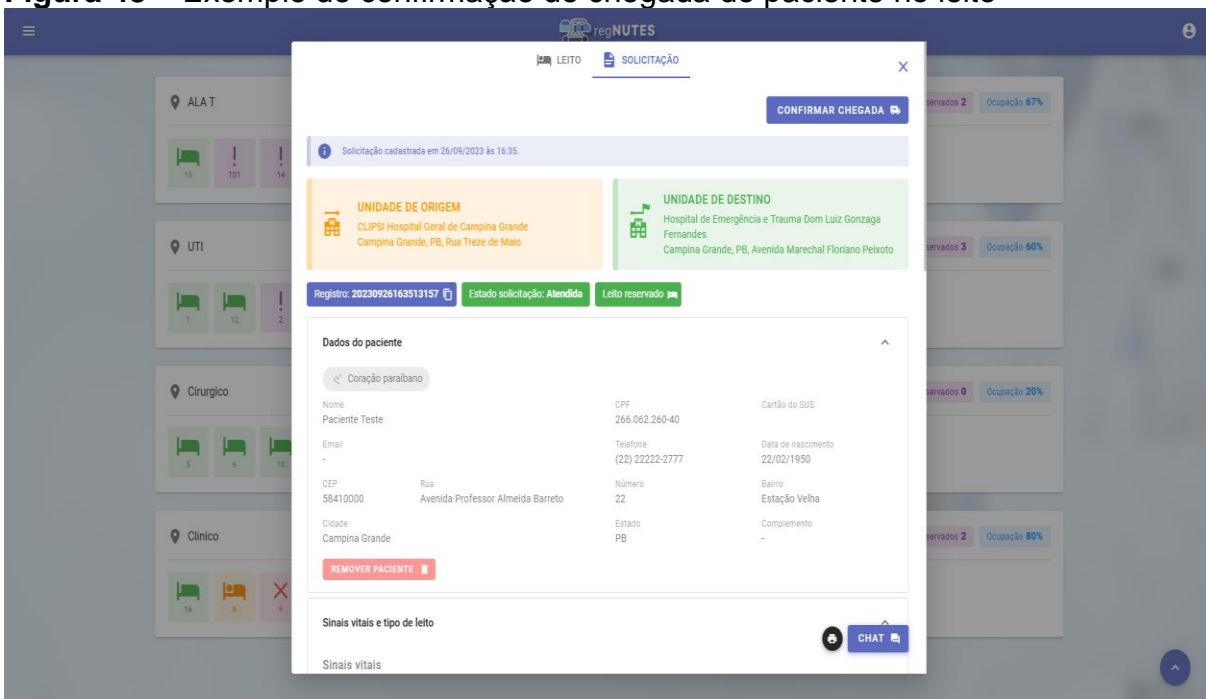
Figura 47 – Exemplo de confirmação de ciência sobre a solicitação do leito



Fonte: Elaborado pelo autor, 2024.

Somente após essa confirmação é que poderá ser autorizado o transporte/transferência do paciente até a Unidade de Saúde. Por fim, a chegada do paciente deve ser confirmada para que ele possa ocupar o leito reservado, clicando no botão observado na Figura 48.

Figura 48 – Exemplo de confirmação de chegada do paciente no leito

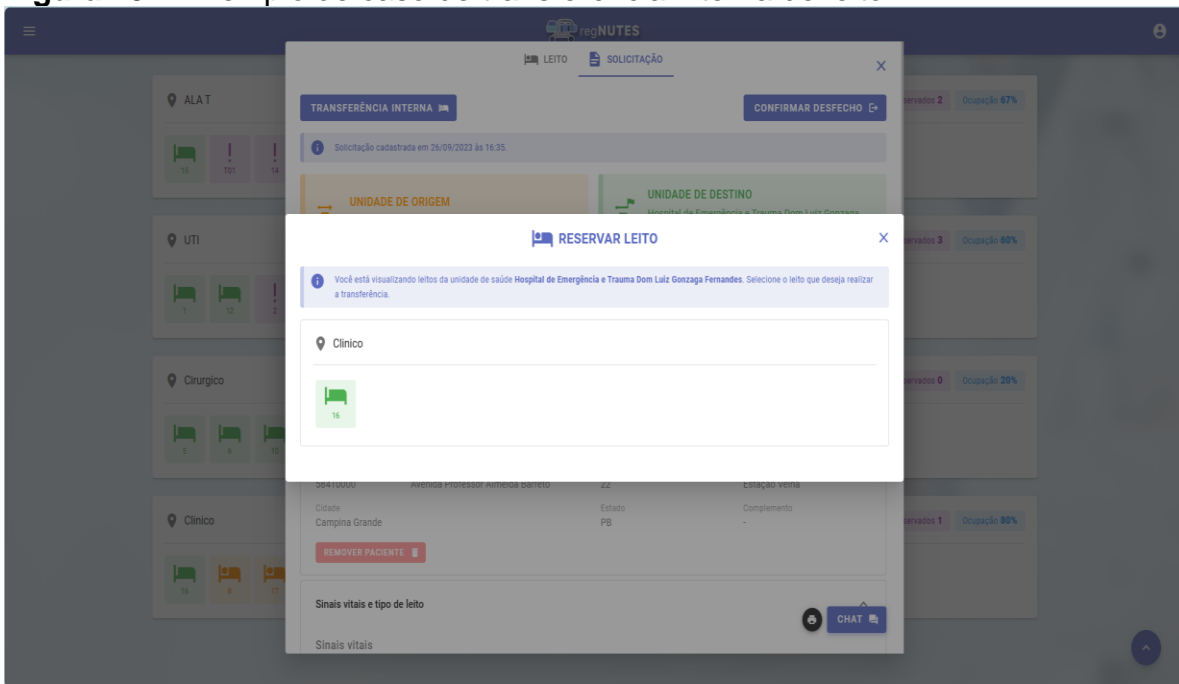


Fonte: Elaborado pelo autor, 2024.

3.6.2.5 Transferência interna

Se houver qualquer problema interno relacionado ao uso do leito, como cama quebrada, problemas na estrutura, etc., pode ser realizada a transferência interna do paciente para outro leito vago, proporcionando mais flexibilidade no uso dos leitos. Um exemplo desse recurso pode ser visto na Figura 49.

Figura 49 – Exemplo de caso de transferência interna de leito



Fonte: Elaborado pelo autor, 2024.

3.6.2.6 Confirmação de desfecho

Caso o paciente já tenha finalizado sua estadia, o desfecho pode ser registrado. Nesse processo, os detalhes finais sobre a liberação do paciente são informados, incluindo o resultado final do tratamento, cujos dados são preenchidos no formulário exibido na Figura 50, para que, se necessário, essas informações possam ser consultadas posteriormente.

Figura 50 – Exemplo de formulário para despecho

The image shows a screenshot of a web application interface. A modal window titled "Confirmação de despecho" is open, displaying a form for selecting a discharge type and providing a justification. The background shows a patient record for "Coração parabaiano" with various tabs and data fields.

Confirmação de despecho

Tipo de despecho*

- Alta Curado
- Alta Melhorado
- Alta a pedido
- Alta com previsão de retorno
- Alta por Evasão
- Alta por outros motivos
- Transfêrencia
- Declaração de óbito fornecida pelo médico
- Declaração de óbito fornecida pelo IML
- Declaração de óbito fornecida pelo SVD

Justificativa

CANCELAR SALVAR ✓

Fonte: Elaborado pelo autor, 2024.

3.6.3 Outros fluxos

Além desses fluxos principais, existem outros que complementam e tornam o fluxo principal completo, os quais são descritos nos subtópicos abaixo.

3.6.3.1 Cadastro de usuários

Um dos fluxos que mais tem influência sobre os fluxos de cirurgia e leito, é o de usuários, pois é nesse fluxo que ocorre a integração de todos os atores do sistema, quais sejam:

- a) Pacientes: são os atores mais utilizados no fluxo principal e as informações que eles disponibilizam são necessárias durante todo o processo para, por exemplo, remarcação de cirurgias, confirmação de entrada, confirmação de dados pessoais, etc.;
- i) Vinculação a uma Unidade de Saúde: Pacientes podem ser vinculados a uma Unidade de Saúde. Anteriormente, essa funcionalidade era utilizada para autorizar operadores a cadastrar as solicitações de leitos ou cirurgias desses pacientes e ver suas informações. Porém, atualmente, essa funcionalidade foi removida;

- ii) Vinculação a uma Secretaria de Saúde: Pacientes podem ser vinculados a uma Secretaria de Saúde. Antes, essa função era usada para autorizar o gestor do referido órgão, bem como os operadores vinculados a cadastrar suas solicitações de leitos ou cirurgias e ver suas informações. Porém, atualmente, esta função também foi removida, perdendo, assim, sua importância no sistema.
- b) Operadores: são responsáveis pela maior parte da interação do fluxo, podendo ser divididos em dois tipos: solicitante e executante. Ressalte-se que o usuário que é cadastrado como operador pode acessar o sistema tanto como solicitante quanto como executante, o que o torna, assim, o ator que possui a maior quantidade de casos de uso vinculados a ele;
 - i) Vinculação a uma Unidade de Saúde: Operadores podem ser vinculados a uma Unidade de Saúde, o que, comumente, os permite ter acesso às solicitações presentes naquela unidade, bem como criá-las ou recebê-las para agendamento e validação. Além disso, é possível controlar os acessos aos fluxos principais do sistema disponíveis ao operador selecionado;
 - ii) Vinculação a uma Secretaria de Saúde: Operadores podem ser vinculados a uma Secretaria de Saúde, mormente para que realizem cadastros e visualizações de solicitações em nome do órgão a que estiver vinculado.
- c) Reguladores: são, na maioria das vezes, médicos e sua função é validar, rejeitar e definir o parecer médico à solicitação. Importante salientar que, quando os reguladores não são médicos, suas funcionalidades são limitadas;
- d) Gestores reguladores: exercem as mesmas funções de um regulador, mas promovem também o gerenciamento dos demais reguladores e a finalização dos processos;
- e) Gestor de Unidade de Saúde: exerce a função de gerência da Unidade de Saúde, de modo que é responsável por todo o fluxo, atuando, por exemplo, na gerência das cirurgias disponíveis e de suas quantidades e no gerenciamento das alas no hospital e dos leitos. Ressalte-se, contudo, que o gestor da Unidade de Saúde não pode definir os status dessas alas;
- f) Gestor de Secretaria de saúde: em regra, exerce a função de cadastro das solicitações, dos pacientes e dos operadores;

- g) Gestor: é um usuário que desempenha um papel relativamente pequeno no fluxo do sistema. Embora tenha algumas funções importantes, como o cadastro de usuários gestores, a criação de Unidades de Saúde, a edição de informações sobre os municípios e a definição das cirurgias que podem ser usadas por outros usuários (como gestores de unidades que as adicionam aos seus hospitais), sua participação no sistema é limitada. Na prática, muitas de suas funções são assumidas pelo administrador, tornando o gestor um usuário substituível no envolvimento ativo no sistema;
- h) Administrador: é, sem dúvida, o usuário com o maior poder de gerenciamento no sistema. Ele tem a capacidade de cadastrar todos os usuários, criar Unidades de Saúde, editar informações das Secretarias de Saúde, e disponibilizar as cirurgias que podem ser usadas no sistema. No entanto, ao contrário dos outros usuários, ele não pode cadastrar solicitações.

3.6.3.2 Cadastro de unidades de saúde

O cadastro das Unidades de Saúde é uma função atribuída ao administrador e ao gestor do sistema, os quais possuem autoridade para criar Unidades e editar as informações existentes.

Esse processo é essencial para garantir que as Unidades estejam corretamente configuradas no sistema, permitindo que sejam selecionadas para reserva de leitos e agendamentos de cirurgias.

As Unidades cadastradas devem incluir informações detalhadas, como endereço, laboratórios, farmácias, as capacidades disponíveis e dados de contato.

3.6.3.3 Autorização do uso das cirurgias

A autorização de uso das cirurgias no sistema funciona como uma *whitelist*, na qual o administrador e o gestor decidem quais cirurgias podem ser autorizadas e realizadas. Esse processo garante que apenas os procedimentos previamente aprovados estejam disponíveis para os usuários.

Importante ressaltar que, caso uma cirurgia seja removida da lista, ela pode ser facilmente reintegrada ao sistema, garantindo flexibilidade na gestão dos procedimentos disponíveis.

3.6.3.4 Estatísticas de cirurgias e leitos

O sistema também permite a gestão e monitoramento de estatísticas relacionadas a cirurgias e leitos. Essas estatísticas incluem o número de cirurgias realizadas, os tipos de procedimentos, a taxa de ocupação dos leitos e outros dados relevantes.

Essas informações são essenciais para a tomada de decisões estratégicas, como a alocação de recursos, o planejamento de capacidade e a melhoria contínua dos serviços de saúde. As estatísticas podem ser usadas tanto para análises internas quanto para a criação de relatórios que informam aos gestores e às Secretarias de Saúde sobre o desempenho das Unidades de Saúde e a eficiência da integralidade do sistema.

4 DESENVOLVIMENTO

Para que o trabalho pudesse ser desenvolvido, foi necessário no primeiro momento, entender as motivações para a criação da documentação do *software* RegNUTES. É possível afirmar que a documentação é extremamente importante no processo de desenvolvimento. Por meio dela que a evolução do *software* é registrada para que sejam criadas as bases necessárias para as possíveis etapas posteriores, as tornando mais simples e práticas, trazendo assim uma maior facilidade no desenvolvimento, treinamento e manutenção de sistema (Nunes; Soares; Falbo, 2011). Pode-se dizer que, a necessidade também advém das diversas funcionalidades do sistema, a compreensão básica das etapas até o resultado, é extremamente necessário para que o sistema possa ser mantido de forma adequada e caso exista a necessidade de uma mudança, que possa ser feita de forma simples e bem orientada.

Após a compreensão das motivações, foi necessário a busca de como fazer desenvolver uma documentação adequada de um *software*. Essa primeira etapa, foi composta de forma majoritária por pesquisas bibliográficas, sendo formada por livros voltados para a parte de engenharia de *software*, como “Software Engineering at Google Lessons Learned from Programming Over Time” de Hyrum Wright, Tom Manshreck, Titus Winters, livros de conteúdo sobre UML, como “UML 2 – Uma abordagem prática” de Gilleanes T. A. Guedes, e também livros buscando o entendimento do uso de documentação em gerência de projetos, com “O Gestor de Projetos” de Richard Newton. Além disso, referências de artigos também foram buscadas, como estudos que destacam as documentações mais usadas para a manutenção de *software* (Souza; Anquetil; Oliveira, 2005), que abordam o custo, o benefício e a qualidade das documentações de desenvolvimento de *software* (Zhi; Garousi; SUN; Garousi; Shahnewaz; Ruhe, 2014), e também, que buscam dizer a importância da documentação para manutenção de *software* (Nunes; Soares; Falbo, 2011). Esses conteúdos tiveram a principal função contextualizar de forma geral o caminho que foi utilizado para o resultado final, sendo eles amplamente utilizados desde o mapeamento das informações do *software* até na escolha dos diagramas.

Com a noção estabelecida de como desenvolver, o passo seguinte se baseia em analisar e encontrar as informações necessárias para o mapeamento. Um dos maiores problemas encontrados nessa etapa foi justamente não haver uma documentação textual técnica, a maioria das mudanças feitas, foram trabalhadas a partir do código fonte, dos

comentários presentes nesse código, no manual do usuário e no conhecimento dos desenvolvedores sobre os fluxos. Porém, a maioria das fontes ainda podem ser consideradas documentações, como o próprio comentário de código, que explica como as funcionalidades trabalham, mesmo que de forma básica (Wright; Manshreck; Winters, 2020, p. 185). Entretanto, outra problemática surge a partir do momento que buscamos o código como referência, mesmo que ele seja ideal para entender com totalidade, ainda existe a necessidade de um conhecimento técnico prévio para interpretar as motivações, e explicar bem do porquê elas serem trabalhadas daquela forma, causando assim uma interpretação não tão aprofundada e por consequência, uma documentação mais básica. Além disso, outro ponto negativo trazido por ele é o quesito de tempo, tanto que existem estudos (Pfleeger, p. 475; Pigoski, p. 35) reportando que de 40% a 60% do tempo gasto na manutenção, pode acabar sendo usado para o estudo do software, sendo para entendê-lo ou para planejar como a mudança pode ser aplicada. Mesmo assim, todas as fontes de informação ainda se mantêm legítimas para a produção dos diagramas, elas contêm, noções das ações presentes no software e também os impactos que cada ação possui, assim é possível dizer que mesmo tendo fontes mais técnicas e que necessitam de um conhecimento da área, ainda é possível utilizá-las no processo.

A partir da obtenção das informações disponíveis sobre o software, é possível seguir para o próximo quesito, sendo ele focado na escolha das documentações que serão trabalhadas e as motivações para seu uso. Após o mapeamento, veio a necessidade de analisar novamente tudo o que foi obtido, buscar no material informativo e nos meios bibliográficos, quais seriam os melhores diagramas para o trabalho. Anteriormente, já havia sido declarado o uso da UML, como base de documentações, justamente pela sua ampla utilização e também pela sua flexibilidade, podendo ser definido justamente pelo leque de diagramas disponíveis para uso, podendo assim cobrir múltiplos aspectos da aplicação. Porém, para a escolha houve uma necessidade de atingir objetivos específicos nos diagramas, como de dar uma visão geral dos usuários e das funcionalidades, tendo o intuito de mostrar tudo que compõe o projeto, também demonstrar de forma mais detalhada o passo a passo do fluxo, definindo como o sistema trabalha.

Continuando com o quesito de escolha de software, para que os dois pontos possam ser alcançados será necessário dividir as responsabilidades, trazendo assim, um diagrama para o contexto geral, com o intuito de apresentar e explicar de forma parcial o funcionamento e um para o específico, sendo esse, usado para demonstrar o fluxo de forma completa. Nesse quesito, é possível destacar um dos primeiros diagramas como

sendo o de casos de uso, sendo ele aquele que mais se sobressai no mapeamento da visão geral do software, justamente por ele não se perder tanto no aprofundamento e ser mais focado em facilitar a identificação dos diferentes níveis de acesso e nas ações de cada ator. No caso de um diagrama mais especializado, existem dois que poderiam trabalhar bem essa parte, sendo eles o diagrama de atividade e o diagrama de sequência, porque ambos oferecem uma visão clara da execução das ações no sistema. Porém, para esse projeto o mais apropriado será o de atividade, justamente, por ele ser ideal quando se deseja modelar o fluxo de trabalho ou processos dinâmicos dentro de um sistema, oferecendo uma visão clara e sequencial de atividades e decisões. Ao contrário dos diagramas de sequência, que é menos adequado para mapear fluxos completos, pois enfatiza a troca de mensagens entre objetos e não o processo em si, o diagrama de atividade é mais adequado para demonstrar o comportamento geral e o fluxo de controle de um processo, independentemente de quem realiza as ações. Isso o torna uma escolha mais intuitiva para entender o processo de forma ampla, facilitando a identificação de gargalos e melhorias no fluxo de atividades.

Assim, a última etapa tem como objetivo a montagem dos diagramas, trazendo consigo, uma busca de estruturas para a sua construção e também, o processo criação dos padrões que farão parte do resultado final da documentação. A primeira análise foi feita para os casos de uso, porém, mesmo eles sendo bem expositivos no seu objetivo, no primeiro momento houve uma falha na compreensão das suas definições e de como criar a sua estrutura, então para mitigar o processo, e não atrasar o desenvolvimento dos diagramas, foi utilizado a estratégia da criação dos diagramas de atividade primeiro, e essas atividades seriam utilizados para representar os casos de uso, porém, é necessário dizer que uma não é igual a outra, um diagrama de atividade pode ser usado para descrever o comportamento por trás de um caso de uso, mas uma atividade em si não equivale a um caso de uso completo.

Seguindo com o processo de desenvolvimento do diagrama de atividade, para a criação das bases, houve uma busca por referências, com o intuito de mimetizar o processo que acontece no RegNUTES, onde o usuário interage com o *front-end*, o *front* interage com o *back-end*, e vice-versa. O problema se apresentou na visualização da estrutura, onde os exemplos disponíveis pelas referências bibliográficas sobre UML, não possuíam uma boa representação que atendesse as necessidades previstas, para poder seguir. Então, foi criado um molde para representar o processo, a partir de 3 partições de atividade, sendo elas, o usuário, o *front-end* e o *back-end*. Assim, podendo concentrar as

atividades em cada parte de forma coesa, facilitando a leitura e dividindo as necessidades de cada processo, sendo interligadas pelos conectores, para uma maior organização.

Continuando com o diagrama de atividade, é necessário pontuar que mesmo com a base organizada, ainda houve diversas problemáticas que foram encontradas de forma majoritária no decorrer da estruturação do modelo, sendo essa, a parte mais trabalhosa de todo o processo, por justamente colocar a documentação em um ciclo quase infinito de criação, revisão e refatoração. É possível pontuar que uma das maiores causas de atraso e de refatoração da documentação foi na criação de padrões. Os padrões podem ser referidos como, regras impostas aos diagramas com o intuito de evitar incongruências nos fluxos, como o caso de um formulário no *front-end* que se for aberto ele deve ser fechado, ou se uma requisição for enviada para o *back-end*, deve ser avaliado se a chave de autorização está válida ainda, se não estiver deve executar uma ação de refazer o login ou parar o fluxo. Porém, esse ciclo de criar padrões necessitava de atualização dos anteriores, ou seja, se fosse definido que quando fosse relogado a página deveria ser reiniciada, esse mesmo padrão iria para todos, ocasionando assim em um *loop* sem fim, de analisar e retrabalhar. Pode-se dizer que a maior dificuldade presente foi na construção do diagrama de atividade, pois, além dos problemas citados, alguns fluxos também não faziam tanto sentido, por não serem mais utilizados hoje ou por serem substituídos por necessidade do cliente, um exemplo é a vinculação do paciente a uma unidade ou secretaria de saúde, que antes permitia somente aos atores vinculados aquelas unidades acessar as informações do paciente, onde hoje, isso não tanto influencia. Foram essas pequenas mudanças constantes que auxiliaram para o aumento do tempo necessário para o desenvolvimento da documentação, atrasando e por fim, não permitindo que o diagrama de atividade fosse finalizado de forma completa.

Entretanto, mesmo com o fluxo do diagrama de atividade não sendo finalizado completamente, graças à construção dos processos principais do sistema, foi possível ter uma noção melhor de como o diagrama de casos de uso funciona. Assim, para criar a base, foi usada a referência presente no livro “UML 2: uma abordagem prática” de Gilleanes T. A. Guedes (2011, p. 84), não tendo tanta diferença com relação à organização, porém, para compor as fronteiras do sistema foram utilizados realmente os fluxos, sendo divididos em quatro, solicitações de cirurgia, solicitações de leitos, unidade de saúde e secretária de saúde. Mesmo com tão poucas fronteiras, foi possível evidenciar boa parte de como o sistema é usado, quais atores têm quais funcionalidades, tudo de forma clara e objetiva, demonstrando até quais casos dependem de quais para executar.

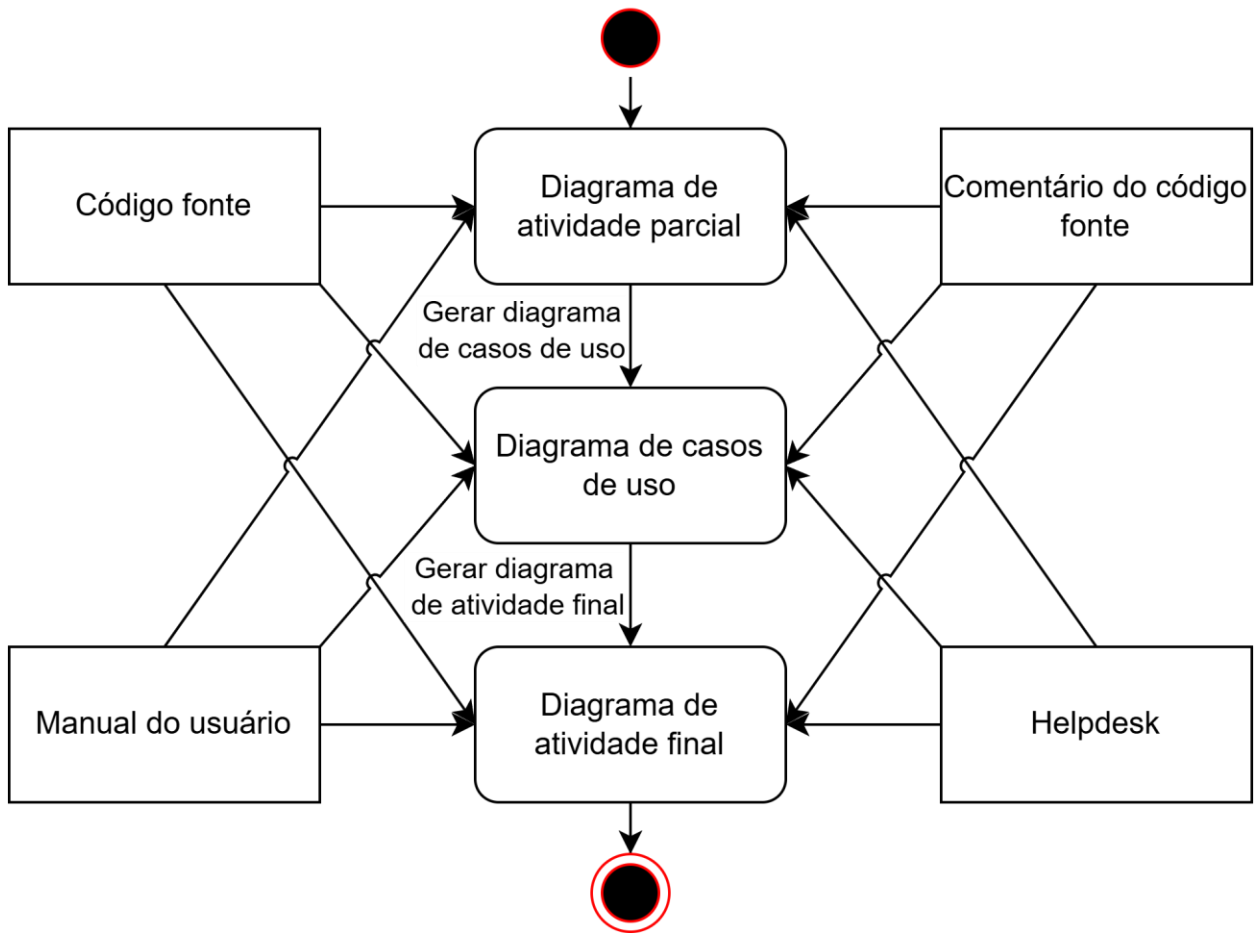
5 RESULTADOS E DISCUSSÕES

Nesta seção, serão apresentados os resultados obtidos com o mapeamento dos processos no sistema RegNUTES, focando nas funcionalidades de solicitação de cirurgias e leitos. O mapeamento foi realizado com base na modelagem de diagramas de Caso de Uso e Atividade, permitindo uma visão detalhada das interações e dos fluxos do sistema. Também será discutido como foi conduzido o processo de mapeamento, incluindo os desafios encontrados na documentação e na modelagem dos fluxos. A seguir, são detalhados os resultados finais referentes aos fluxos de cirurgias e leitos, e a forma como esses fluxos foram aprimorados.

5.1 Mapeamento e construção

O mapeamento dos processos foi uma etapa essencial para garantir a documentação precisa e a compreensão clara de todas as funcionalidades e fluxos do sistema. Esse trabalho incluiu a análise minuciosa de múltiplas fontes, como o código-fonte, os comentários inseridos no código, o manual do usuário e as interações com o *helpdesk*, com o objetivo de construir diagramas que retratassem com precisão as atividades e interações dos usuários no sistema. A partir dessas análises, foi possível criar e refinar tanto os diagramas de atividade quanto os de casos de uso, que foram fundamentais para assegurar uma documentação clara e coerente. A Figura 51 apresenta uma representação desse processo e suas influências. Nos subtópicos a seguir, são detalhados os passos adotados para obter as informações e como elas contribuíram para a criação dos documentos finais.

Figura 51 – Mapa do mapeamento da criação da documentação



Fonte: Elaborado pelo autor, 2024.

5.1.1 Código fonte e comentários de código

A análise do código-fonte foi o ponto de partida essencial para o mapeamento do sistema, fornecendo uma visão clara da lógica implementada e das interações entre os diferentes componentes. Junto com os comentários presentes no código, essa etapa permitiu compreender o funcionamento de funções e módulos, facilitando a identificação de pontos chave no fluxo de atividades e processos do sistema. A partir dessa base, foram extraídas informações relevantes que subsidiaram a criação inicial dos diagramas, ajudando a esclarecer a estrutura e o comportamento esperado do sistema.

Para que a análise pudesse ser feita, foi iniciado com o *front-end* como primeiro passo, justamente por ele conter as primeiras interações com o usuário, foi possível usá-la na criação do diagrama de atividade parcial. Nesse processo, primeiro foi necessário analisar as interações do sistema, normalmente usadas pelos usuários para usar as funcionalidades, sendo constituída de componentes (botões, campos de entrada, janelas,

entre outros), e com essa análise, buscar um referencial nas linhas de código, sempre comparando alguma informação presente na página e relacionando com o código que as cria. Tendo essa noção básica, a seleção das informações para representação deve ser feito por meio de critérios, como o caso de validações, elas devem ser sempre representadas pois, podem alterar como fluxo do sistema funciona, ou também botões, pois eles iniciam, dão prosseguimento ou finalizam um processo, trazendo assim, uma importância no seu uso e na sua representação.

A análise do *front-end* foi essencial para entender as interações iniciais do usuário, como formulários preenchidos e botões clicados. Por meio dele, o *back-end* pode ser mapeado, para assim, demonstrar como essas interações geram mudanças nos dados, como a criação ou atualização de registros. Para esse mapeamento foi necessário, seguir um caminho menos profundo, justamente pela falta de experiência no campo, foi necessário entender primeiro o funcionamento da arquitetura de micro serviços e como os códigos separados interagem. O principal foco da análise foi na procura de nomes de funções ou funcionalidades que fossem chamadas no decorrer fluxo, sendo desde a chegada da requisição HTTP até o envio da resposta, ou de comentários que explicassem as suas aplicabilidades no sistema. Nas análises foram focados só em informações que afetariam o resultado final, como validar informações ou alterar status, e também ações que pudessem ser notadas no *front*, como no caso de alteração de dados.

5.1.2 Manual do usuário e helpdesk

Além do código, o manual do usuário e as interações com o *helpdesk* desempenharam um papel fundamental para a documentação dos fluxos operacionais. O manual forneceu uma visão prática e acessível das funcionalidades do sistema, enquanto o *helpdesk* ajudou a identificar problemas recorrentes e dúvidas dos usuários, oferecendo *insights* valiosos sobre como os processos eram utilizados na prática. Esses dois recursos complementaram a análise técnica, trazendo uma perspectiva mais próxima da experiência do usuário final e ajudando a ajustar os diagramas de atividade de acordo com as necessidades reais dos usuários.

5.1.3 Diagrama de atividade parcial

Com as informações coletadas do código fonte, dos comentários, do manual do usuário e das interações com o *helpdesk*, foi possível construir um diagrama de atividade

parcial. Esse primeiro esboço mapeou as principais atividades do sistema, sempre focando na divisão dos três campos de execução dos processos, *back-end*, *front-end* e o usuário, mas ainda necessitava de refinamentos para cobrir fluxos mais complexos ou exceções. O diagrama de atividade parcial serviu como um rascunho, permitindo que *fosse tendo uma visão clara* sobre as etapas principais e que fossem feitos ajustes conforme as necessidades do projeto.

5.1.4 Diagrama de casos de uso

Após a construção do diagrama de atividade parcial, foi gerado o diagrama de casos de uso. Esse diagrama destacou as interações entre os diferentes atores e o sistema, representando de forma clara e objetiva as funcionalidades oferecidas. A criação do diagrama de casos de uso auxiliou na identificação de lacunas ou redundâncias no sistema, além de facilitar o entendimento sobre as responsabilidades e permissões de cada ator envolvido.

5.1.5 Diagrama de atividade final

Com o refinamento das informações e a incorporação de *feedbacks* oriundos dos diagramas parciais e de casos de uso, o diagrama de atividade final foi gerado. Este diagrama apresentava uma visão completa e detalhada de todos os fluxos do sistema, incluindo variações, fluxos alternativos e exceções. Ele serviu como a principal referência para o desenvolvimento, garantindo que as atividades fossem implementadas de maneira coesa e em conformidade com os requisitos do sistema. A versão final do diagrama de atividade, portanto, consolidou todo o trabalho de análise e documentação realizado ao longo do processo.

5.2 Cirurgias

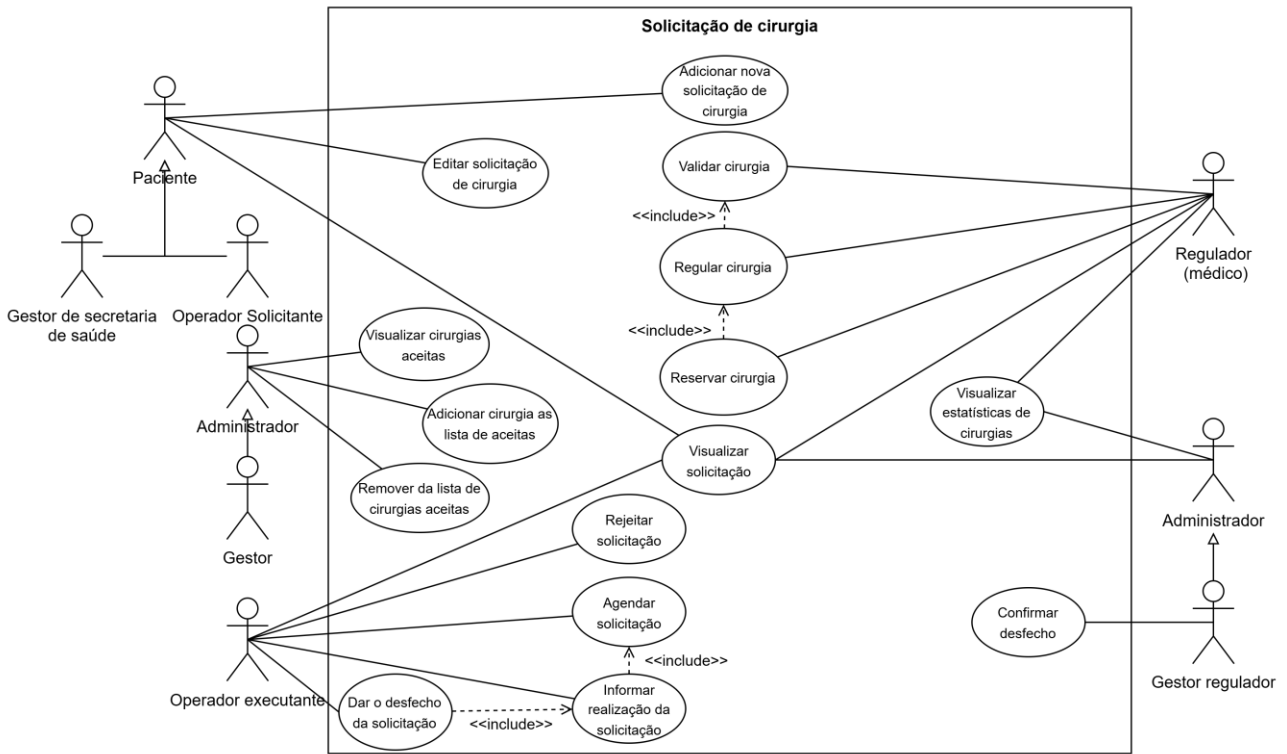
A modelagem do fluxo de cirurgias foi uma parte central do desenvolvimento, visando mapear todas as etapas que envolvem a solicitação, agendamento e acompanhamento de procedimentos cirúrgicos. A partir das informações obtidas por meio da análise do sistema, o diagrama de casos de uso foi construído para representar as interações entre usuários e o sistema, enquanto o diagrama de atividades foi utilizado para

detalhar os processos internos, como validação de dados e autorização de cirurgias. Abaixo, explicaremos os dois diagramas elaborados.

5.2.1 Diagrama de casos de uso: Cirurgias

O diagrama de casos de uso para cirurgias, representado na figura 52, escreve as interações entre os principais atores e o sistema. Aqui, o operador é o responsável por solicitar uma nova cirurgia, que, por sua vez, passa por uma validação e autorização realizadas por diferentes níveis de gestores e reguladores. Este diagrama também inclui casos de uso como o agendamento da cirurgia, a confirmação da solicitação e o acompanhamento do processo por parte do paciente e dos administradores. Esse mapeamento é essencial para garantir que todos os fluxos envolvidos na gestão de cirurgias estejam claros e bem definidos.

Figura 52 – Diagrama de casos de uso do fluxo de cirurgias



Fonte: Elaborado pelo autor, 2024.

5.2.1.1 Descrição geral do diagrama

Essa é a representação da modelagem das operações necessárias para gerenciar solicitações de cirurgia, incluindo desde a adição de novas solicitações até a sua validação e agendamento. Ilustra, assim, como diferentes atores interagem no sistema para garantir que as cirurgias sejam solicitadas, avaliadas e processadas de forma adequada.

5.2.1.2 Atores envolvidos

Os atores envolvidos na interação com o sistema são os seguintes:

- Paciente: inicia o processo ao solicitar uma cirurgia, seja diretamente ou através de um operador solicitante;
- Operador Solicitante: atua no registro das solicitações de cirurgia e na interação com outros atores para garantir que as solicitações sejam processadas corretamente;
- Gestor da Secretaria de Saúde: encarregado de solicitar e supervisionar que as cirurgias sejam realizadas conforme as normas estabelecidas;
- Operador Executante: responsável por executar ações relacionadas ao agendamento e processamento das solicitações, incluindo rejeitar solicitações, quando necessário;
- Regulador (Médico): regula as cirurgias, avaliando a necessidade e a viabilidade das solicitações feitas;
- Gestor Regulador: cuida de supervisionar solicitações e também de fazer o desfecho imediato, caso exista algum problema no processo de reserva ou agendamento;
- Administrador: usuários que gerenciam o sistema, porém não possuem ações que alteram o curso de um fluxo. Suas funções são mais focadas no cadastro dos demais atores.

5.2.1.3 Principais casos de uso

Os principais casos de uso podem ser elencados da seguinte forma:

- Adicionar Nova Solicitação de Cirurgia: o operador solicitante, o gestor de Secretaria de Saúde ou o paciente adicionam as novas solicitações ao sistema, as quais podem ser, posteriormente, validadas e processadas;
- Editar Solicitação: o operador solicitante, o gestor de Secretaria de Saúde ou o paciente têm a capacidade de editar ou rejeitar solicitações de cirurgia, dependendo do contexto;
- Validar Cirurgia: o regulador (médico) verifica se a solicitação de cirurgia atende aos critérios necessários para prosseguir;
- Regular Cirurgia: envolve a avaliação contínua das solicitações para garantir que estão em conformidade com as diretrizes médicas e administrativas;
- Reservar Solicitação: o regulador pode reservar vaga em um hospital que possui disponibilidade de realização daquela cirurgia;
- Rejeitar Solicitações: o operador executante pode rejeitar a reserva da cirurgia, caso exista algum problema ou pendência na solicitação;
- Agendar Solicitação: o operador executante agenda a cirurgia conforme a disponibilidade e a aprovação da solicitação;
- Desfecho da Solicitação: o operador executante pode dar baixa no paciente, informando o status pós-cirúrgico para finalizar a solicitação;
- Visualizar Estatísticas de Cirurgias: o regulador, o gestor regulador ou o administrador podem acessar dados estatísticos sobre as cirurgias, permitindo uma visão ampla dos resultados dos processos;
- Adicionar e remover cirurgias aceitas: o administrador ou gestor podem fazer a gestão no sistema, indicando se determinadas cirurgias podem ser realizadas.

5.2.1.4 Interpretação e impacto no projeto

O diagrama representa, de maneira clara e eficaz, as interações necessárias para gerenciar o fluxo de solicitações de cirurgia no sistema, destacando como cada ator tem um papel específico e interage com o sistema para garantir que as cirurgias sejam devidamente solicitadas, avaliadas e agendadas.

Ressalte-se que a criação do diagrama teve impacto na garantia de que os requisitos funcionais do sistema fossem compreendidos de maneira uniforme entre todos os membros da equipe de desenvolvimento, reduzindo ambiguidades e promovendo uma implementação alinhada com as necessidades do projeto.

5.2.1.5 Discussão

É possível afirmar que o diagrama de casos de uso é uma ferramenta importante, posto que proporciona facilidade de visualização e organização dos processos complexos envolvidos na solicitação de cirurgias, promovendo, assim, uma maior compreensão dos diferentes papéis e das interações críticas dentro do sistema.

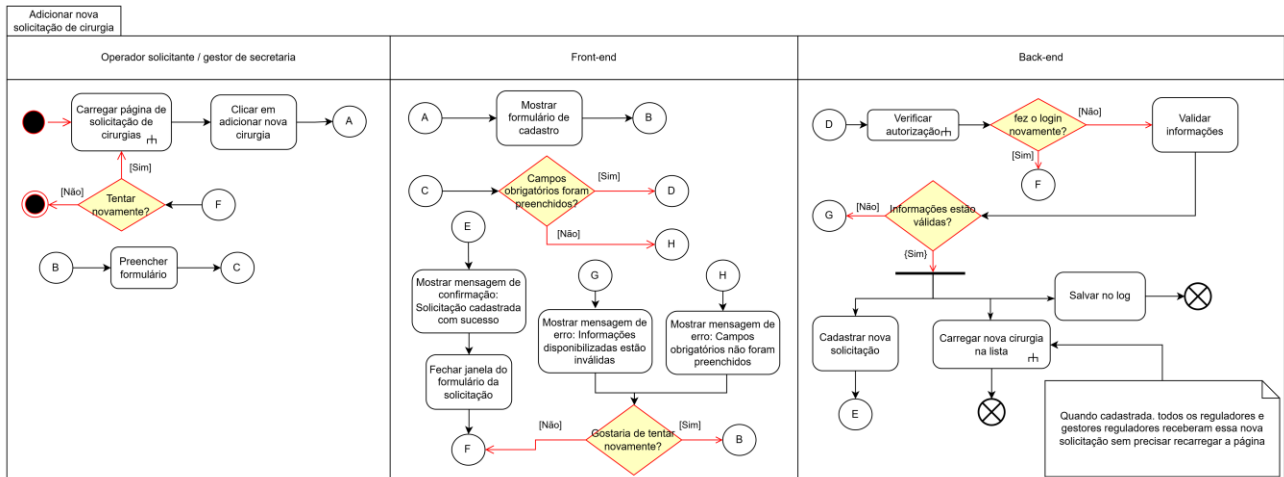
No entanto, a complexidade desse sistema exige também a criação de diagramas complementares para detalhar processos mais específicos, sendo demonstrado, de forma prática, a necessidade de algo suplementar na falta de entendimento das funcionalidades dos casos de uso apresentados.

Apesar disso, a utilização de um único diagrama abrangente, como o apresentado, proporciona uma visão geral. Ademais, a modelagem dos casos de uso contribui significativamente para a comunicação e o aprendizado entre aqueles que buscam compreender o sistema, resultando, assim, em uma base mais eficiente e bem estruturada.

5.2.2 Diagrama de atividade: Cirurgias

No diagrama de atividade, representado na figura 53, é possível visualizar as etapas detalhadas do fluxo de cirurgias, desde a solicitação inicial até a conclusão do procedimento. Cada decisão, como a validação e a autorização da cirurgia, é representada de forma a esclarecer como o sistema processa essas etapas e quais caminhos alternativos podem ser seguidos, como a necessidade de correção de dados ou a recusa de uma cirurgia. Essa modelagem de atividades garante que os processos internos do sistema estejam devidamente documentados, ajudando a identificar possíveis gargalos e oportunidades de otimização.

Figura 53 – Diagrama de atividade do fluxo de adicionar nova solicitação de cirurgia



Fonte: Elaborado pelo autor, 2024.

5.2.2.1 Descrição geral do diagrama

É possível notar que o diagrama de atividade faz a modelagem do processo completo de adição de uma nova solicitação de cirurgia, buscando abranger todas as ações realizadas, desde as interações com o sistema até as validações de informações para garantir o cadastro da solicitação.

5.2.2.2 Atores envolvidos

De forma sintetizada, são os seguintes os autores envolvidos descritos no diagrama de atividade:

- Carregar Página de Solicitação de Cirurgias: o processo se inicia com o operador solicitante acessando a página específica para adicionar uma nova cirurgia, sendo essa atividade uma ação de chamada de comportamento, usada para invocar a execução de outro diagrama;
- Preenchimento e Validação: o operador preenche o formulário de solicitação e, logo após, o sistema verifica se todos os campos obrigatórios foram preenchidos corretamente;
- Mensagens de Confirmação ou Erro: dependendo da validade das informações, o sistema pode exibir uma mensagem de confirmação, completando a solicitação, ou uma mensagem de erro, solicitando correção e reenvio dos dados;

- Processamento no *Back-end*: o sistema realiza uma série de verificações automatizadas, como a validação de autorizações e a atualização de *logs*, antes de finalizar o registro da solicitação.

5.2.2.3 Principais decisões e alternativas

Elenca-se, em sequência, as principais decisões e alternativas do usuário, conforme o diagrama de atividade:

- Tentar Novamente: o diagrama inclui pontos de decisão nos quais o operador pode optar por tentar novamente caso algum erro seja encontrado, como no preenchimento de campos obrigatórios ou na validação de login;
- Salvamento e *Log*: o *back-end* é responsável por salvar a solicitação no sistema e atualizar os registros de *log*, assegurando a rastreabilidade das operações realizadas.

5.2.2.4 Interpretação e impacto no projeto

O presente caso, exibido no diagrama de atividade, proporciona uma visão detalhada e sequencial do processo, essencial para a compreensão das operações internas do sistema e para a garantia de que todos os requisitos de negócio estão sendo atendidos, sendo considerado um complemento do que já foi estabelecido pelo diagrama de casos de uso.

Além disso, esse diagrama é crucial para identificar possíveis gargalos ou áreas que necessitam de melhorias, como a otimização das mensagens de erro para uma melhor experiência do usuário.

5.2.2.5 Discussão

Pode-se dizer que a clareza do fluxo de atividades no diagrama assegura um maior aprofundamento no que foi apresentado antes pelo diagrama de casos de uso, a fim de que fosse possível compreender as etapas necessárias para adicionar uma nova solicitação de cirurgia.

Durante a modelagem, foi preciso considerar diversas alternativas e fluxos de exceção, como a validação de informações incorretas, o que evidenciou a complexidade

do processo. No entanto, a utilização do diagrama de atividade foi crucial para visualizar o processo em diferentes níveis de detalhe, desde a interação do usuário até os procedimentos internos do sistema, garantindo uma implementação consistente e alinhada aos requisitos.

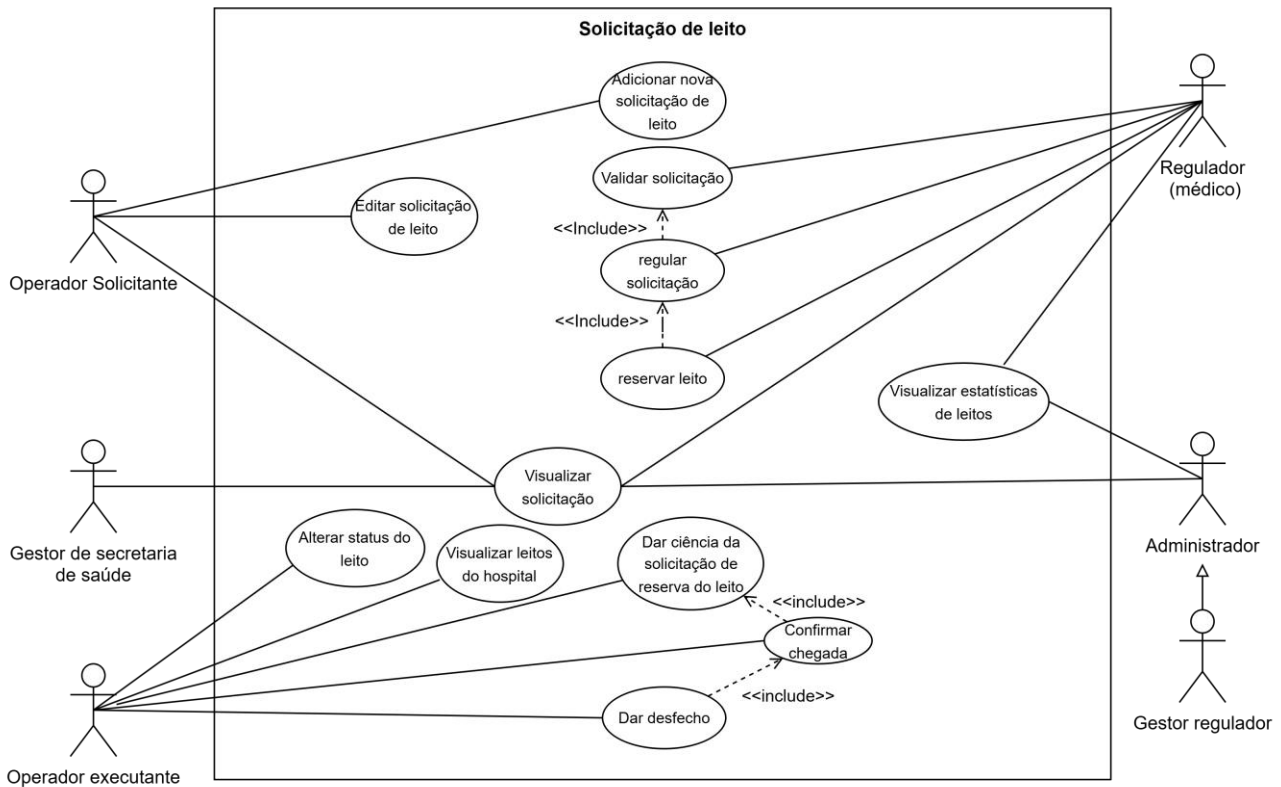
5.3 Leitos

O fluxo de leitos segue uma estrutura complexa que envolve múltiplos atores e processos para garantir a disponibilidade, reserva e acompanhamento de leitos hospitalares. A criação do diagrama de casos de uso e de atividades para esse fluxo permitiu um melhor entendimento das etapas envolvidas e do papel de cada usuário no processo de gestão de leitos. A seguir, apresentamos os diagramas elaborados para este fluxo.

5.3.1 Diagrama de casos de uso: Leitos

O diagrama de casos de uso para o fluxo de leitos, vista na figura 54, destaca as interações dos operadores, gestores e reguladores com o sistema. As principais ações, como a solicitação, a reserva e a atualização de status dos leitos, são detalhadas para garantir que todos os usuários compreendam suas responsabilidades no processo. Esse mapeamento ajuda a evitar problemas de comunicação e falhas no processo de gestão de leitos, assegurando que o sistema funcione de maneira eficiente e integrada.

Figura 54 – Diagrama de casos de uso do fluxo de leitos



Fonte: Elaborado pelo autor, 2024.

5.3.1.1 Descrição geral do diagrama

Este diagrama de atividade ilustra o processo completo de adicionar uma nova solicitação de leito dentro do sistema. Ele detalha o fluxo de atividades desde o carregamento da página até a confirmação de sucesso da solicitação, incluindo a interação entre o *front-end* e o *back-end* para validar e armazenar os dados.

5.3.1.2 Atores envolvidos

Os atores envolvidos no diagrama incluem:

- **Operador Solicitante:** Responsável por iniciar a solicitação de um leito e editar solicitações anteriores.
- **Operador Executante:** Atua na alteração do status do leito e na visualização das informações do hospital.
- **Gestor de Secretaria de Saúde:** Faz a intermediação e atualização do status dos leitos.

- Regulador (médico): Encarregado de validar as solicitações e regular a disponibilidade de leitos.
- Administrador e Gestor Regulador: Responsáveis por visualizar estatísticas e gerenciar as informações sobre os leitos.

5.3.1.3 Principais casos de uso

- Adicionar nova solicitação de leito: Este é o ponto de partida para a maioria das interações. O operador solicitante inicia uma nova solicitação, especificando detalhes relevantes, como o tipo de leito necessário e o paciente em questão.
- Editar solicitação de leito: Caso seja necessário, o operador solicitante pode modificar uma solicitação já existente antes que ela seja processada pelo regulador.
- Validar solicitação: O regulador (médico) é responsável por verificar as informações da solicitação, certificando-se de que todos os requisitos são atendidos antes de permitir que a solicitação siga adiante. Este é um ponto crítico no fluxo, pois a aprovação do regulador é o que permite a continuidade do processo.
- Regular solicitação: Uma vez validada, a solicitação passa para a etapa de regulação, onde o leito solicitado é atribuído e reservado com base na disponibilidade.
- Reservar leito: Após a regulação, o leito é formalmente reservado para o paciente. Este processo garante que o leito será mantido até a chegada do paciente.
- Visualizar solicitação: A todo momento, tanto os operadores quanto os gestores podem consultar o status de uma solicitação, acompanhando seu progresso e verificando se foi aprovada, regulada ou concluída.
- Alterar status do leito: O operador executante pode atualizar o status do leito, alterando sua disponibilidade com base no uso atual ou futuro.
- Confirmar chegada: No final do processo, é necessário que o operador executante confirme a chegada do paciente ao hospital, para que o leito seja efetivamente ocupado.
- Desfecho: O último caso de uso marca o encerramento do ciclo de solicitação, onde o operador registra o desfecho, como alta do paciente ou liberação do leito.

5.3.1.4 Interpretações e impactos no projeto

A criação desse diagrama possibilitou uma visão detalhada das interações e processos dentro do sistema, facilitando o entendimento sobre como os diferentes atores se relacionam com o fluxo de solicitação de leitos. Ele também permitiu identificar gargalos no processo, como a importância da validação por parte do regulador (médico), garantindo que somente solicitações aprovadas sigam adiante. Ao documentar essas interações, o projeto obteve maior clareza na comunicação entre os desenvolvedores e *stakeholders*, assegurando que o sistema pudesse atender às necessidades reais dos usuários.

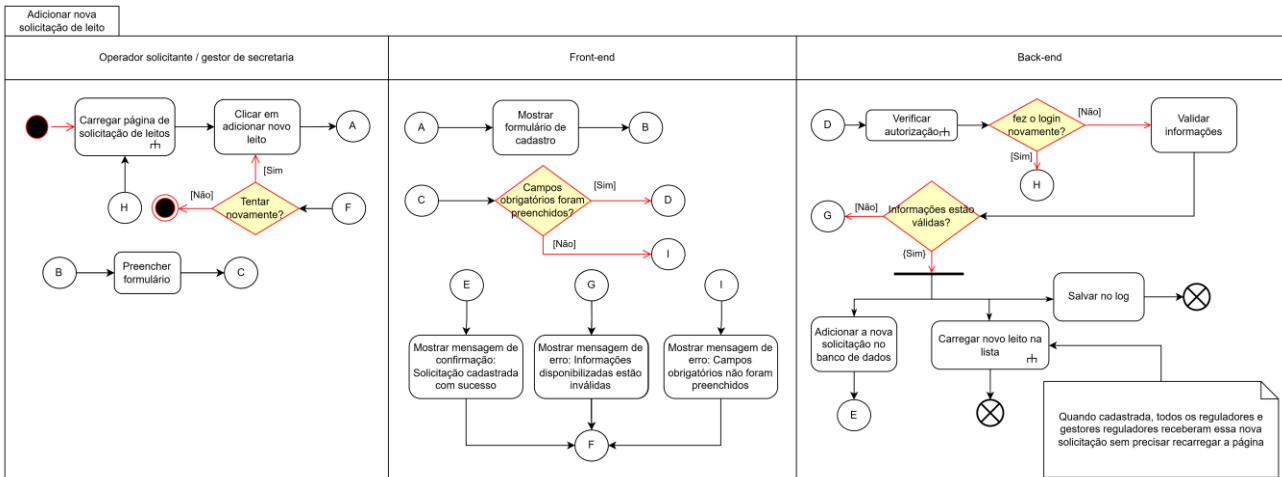
5.3.1.5 Discussões

Durante o desenvolvimento do diagrama, algumas questões relevantes foram levantadas. Uma delas foi a necessidade de criar fluxos de exceção claros, como no caso de uma solicitação inválida ou a falta de disponibilidade de leitos. A visualização desse cenário dentro do diagrama de casos de uso permite que soluções possam ser implementadas para evitar gargalos ou falhas no atendimento. Além disso, a interação entre diferentes tipos de usuários também revelou a necessidade de um sistema de autorização eficaz, onde o papel de reguladores e administradores é essencial para garantir o funcionamento adequado do processo.

5.3.2 Diagrama de atividade: Leitos

No diagrama de atividade para o fluxo de leitos, podendo ser analisada na figura 55, são descritas as etapas internas do sistema, desde a solicitação de um leito até sua reserva e liberação. O diagrama inclui pontos críticos como a validação da solicitação, a disponibilidade de leitos e a comunicação entre os diferentes atores do sistema. Esse nível de detalhamento garante que o sistema esteja preparado para lidar com diversas situações, como lotação de leitos ou cancelamentos, otimizando assim o fluxo de trabalho.

Figura 55 – Diagrama de atividade do fluxo de adicionar nova solicitação de leito



Fonte: Elaborado pelo autor, 2024.

5.3.2.1 Descrição geral do diagrama

Este diagrama de atividade ilustra o processo completo de adicionar uma nova solicitação de leito dentro do sistema. Ele detalha o fluxo de atividades desde o carregamento da página até a confirmação de sucesso da solicitação, incluindo a interação entre o *front-end* e o *back-end* para validar e armazenar os dados.

5.3.2.2 Atores envolvidos

Os atores principais envolvidos neste processo são o Operador Solicitante e o Gestor de Secretaria, que iniciam e gerenciam as solicitações de leitos através do sistema. A dinâmica entre o *front-end* e o *back-end* também desempenha um papel crucial, servindo como intermediários para processar as informações fornecidas.

5.3.2.3 Principais decisões e alternativas

- Verificação de preenchimento correto dos campos: Se os campos obrigatórios não forem preenchidos, o sistema alerta o usuário, evitando o avanço de solicitações incompletas.
- Validação das informações: Antes de adicionar a solicitação ao banco de dados, o *back-end* verifica a autenticidade e a validade dos dados inseridos, assegurando a integridade das informações.

- *Feedback* imediato ao usuário: O sistema fornece mensagens de erro ou confirmação, dependendo do resultado das verificações, mantendo o usuário informado sobre o status da

5.3.2.4 Interpretações e impactos no projeto

A estrutura do diagrama facilita a compreensão dos passos necessários para a solicitação de leitos e realça a importância da validação de dados para manter a eficiência e a confiabilidade do sistema. Este fluxo bem definido pode ajudar a reduzir erros operacionais e melhorar a experiência do usuário ao garantir que todas as solicitações sejam tratadas corretamente.

5.3.2.5 Discussões

As discussões em torno deste diagrama podem se centrar em como otimizar o processo para reduzir atrasos e erros. Por exemplo, poderia ser explorada a possibilidade de uma validação mais automática dos dados ou a implementação de uma interface mais intuitiva para reduzir as chances de erro por parte dos operadores.

6 CONCLUSÃO

Este trabalho teve como objetivo a modelagem e análise de processos críticos em um sistema de gerenciamento de saúde, com foco na criação de diagramas de Caso de Uso e Atividade para ilustrar as interações e os fluxos envolvidos na solicitação de leitos e cirurgias. A modelagem desses processos revelou-se uma ferramenta essencial para garantir a clareza e a eficiência na comunicação entre os desenvolvedores, os *stakeholders* e os usuários do sistema.

A criação dos diagramas de Caso de Uso permitiu uma visão abrangente dos diferentes atores e suas interações dentro do sistema, facilitando a identificação dos principais requisitos funcionais e não funcionais. Esse tipo de modelagem demonstrou ser fundamental para estruturar as funcionalidades do sistema de maneira coerente, assegurando que todas as operações essenciais fossem contempladas e bem definidas.

Por outro lado, o diagrama de atividade proporcionou uma compreensão detalhada do fluxo de trabalho, evidenciando as etapas necessárias para a realização de processos complexos, como a solicitação de cirurgias. A clareza desse fluxo não só facilitou a comunicação entre a equipe de desenvolvimento e os *stakeholders*, mas também garantiu que todos os possíveis cenários, inclusive os de exceção, fossem devidamente considerados durante o desenvolvimento do sistema.

Apesar dos desafios enfrentados, como a necessidade de considerar várias alternativas e fluxos de exceção, a modelagem realizada trouxe inúmeros benefícios para o projeto. A utilização dos diagramas permitiu uma visualização detalhada dos processos, o que foi crucial para garantir uma implementação consistente e alinhada aos requisitos definidos. Além disso, essa abordagem contribuiu significativamente para a melhoria da estrutura do sistema, tornando-o mais robusto e eficiente.

Para trabalhos futuros, uma importante área a ser explorada envolve o aperfeiçoamento dos fluxos não finalizados, especialmente os diagramas de atividade, que enfrentaram desafios devido à criação e adaptação de padrões. Outro ponto seria expandir a documentação para abranger novas funcionalidades que possam ser implementadas no RegNUTES, como a integração de novos módulos ou a adaptação às mudanças nas necessidades dos usuários.

Em síntese, a modelagem dos processos utilizando os diagramas de Caso de Uso e Atividade foi fundamental para o sucesso deste projeto. Ela assegurou uma comunicação clara e eficaz entre todos os envolvidos, permitiu a identificação de pontos críticos e

potencializou a qualidade do sistema desenvolvido. A aplicação dessas técnicas de modelagem, portanto, demonstra-se essencial em projetos de desenvolvimento de sistemas complexos, como o RegNUTES, contribuindo para a entrega de soluções que atendem às necessidades dos usuários de forma eficiente e confiável.

REFERÊNCIAS

- BLOCH, M.; BLUMBERG, S.; LAARTZ, J. **Delivering large-scale IT projects on time, on budget, and on value**. McKinsey Quarterly, [s. l.], v. 27, p. 2-7, 2012.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified modeling language user guide**. 2. ed. London: Addison-Wesley, 2005.
- FOWLER, M. **UML distilled**: a brief guide to the standard object modeling language. 3. ed. London: Addison-Wesley, 2004.
- GILLEANES, T. A. G. **UML 2**: uma abordagem prática. 2. ed. São Paulo: Novatec, 2011.
- GITHUB. **GitHub**. Disponível em: <https://github.com>. Acesso em: 12 ago. 2024.
- LARMAN, C. **Applying UML and patterns**: an introduction to object-oriented analysis and design and iterative development. 3. ed. Pearson Education: London, 2005.
- MICROSOFT. **Visual Studio Code**. Disponível em: <https://code.visualstudio.com>. Acesso em: 12 ago. 2024.
- NEWTON, Richard. **O gestor de projetos**. 1. ed. São Paulo: M. Books, 2008.
- NUNES, V. B. **Apoio à documentação em um ambiente de desenvolvimento de software**. 2011. Disponível em: https://nemo.inf.ufes.br/wp-content/papercite-data/pdf/apoio_a_documentacao_em_um_ambiente_de_desenvolvimento_de_software_2004.pdf. Acesso em: 11 dez. 2023.
- OMG. **Unified Modeling Language (UML) Specification**. 2021. Disponível em: <https://www.omg.org/spec/UML>. Acesso em: 19 abr. 2024.
- PFLEEGER, S. L. **Software engineering**: theory and practice. 2. ed. Prentice Hall, 2001.
- PIGOSKI, T. M. **Practical software maintenance**: best practices for software investment. John Wiley & Sons, Inc., 1996.
- RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. **The unified modeling language reference manual**. 2. ed. London: Addison-Wesley, 2005.
- SEIBERT, F. **Draw.io**: Online Diagram Software. 2023. Disponível em: <https://www.draw.io>. Acesso em: 11 ago. 2024.
- SOUZA, S. C. B.; ANQUETIL, N.; OLIVEIRA, K. M. **A study of the documentation essential to software maintenance**. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON DESIGN OF COMMUNICATION: DOCUMENTING & DESIGNING FOR PERVASIVE INFORMATION, 2005, New York. Proceedings [...]. New York: ACM, 2005. p. 68–75.

TITUS, Titus Winters; MANSFIELD, Tom; WRIGHT, Hyrum. **Software engineering at Google: lessons learned from programming over time**. 1. ed. Sebastopol: O'Reilly Media, 2020.

ZHI, J.; GAROUSI, V.; SUN, B.; GAROUSI, G.; SHAHNEWAZ, S.; RUHE, G. Cost, **benefits and quality of software development documentation**: a systematic mapping. *Journal of Systems and Software*, v. 99, p. 42–54, 2014. DOI: 10.1016/j.jss.2014.09.042.