UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT
DEPARTAMENTO DE INFORMÁTICA – DI

Luana Dantas Pontes Espínola Casado

M Labs: Sistema Para Gerenciamento de Laboratórios

MOSSORÓ - RN
2022

Luana Dantas Pontes Espínola Casado

M Labs: Sistema Para Gerenciamento de Laboratórios

Monografia apresentada à Universidade do Estado do Rio Grande do Norte como um dos pré-requisitos para obtenção do grau de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Maximiliano Araujo da Silva Lopes.

MOSSORÓ - RN

2022

Luana Dantas Pontes Espínola Casado

M Labs: Sistema Para Gerenciamento de Laboratórios

Monografia apresentada como pré-requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovado em: __ / __ / _____

Banca Examinadora

_____

Dr. Maximiliano Araujo da Silva Lopes (Orientador(a))
Universidade do Estado do Rio Grande do Norte – UERN

_____

Membro Examinador 1
Universidade do Estado do Rio Grande do Norte – UERN

_____

Membro Examinador 2
Universidade do Estado do Rio Grande do Norte – UERN

Dedicatória

Dedico este trabalho à minha família que me apoiou até este momento.

AGRADECIMENTOS

Agradecer primeiramente à minha família, na qual me apoiou e foi o ponto principal para conseguir alcançar todos os meus objetivos.

Ao meu avô Francisco Espínola, o homem da minha vida, quem me deixou seu legado, sua história de vida e quem sempre lutou por mim e pelo meu bem estar, à minha avó Adna Espínola que sempre me inspirou em suas histórias de luta como mulher, me ensinou sobre doação e amor.

À minha mãe Nadja Espínola e meu padrasto Raniere Fernandes, quem me deram meus maiores conselhos, me guiaram no meu maior trajeto conhecido como a Vida e me tornaram a pessoa que sou hoje, meu amor por vocês é infinito. Ao meu irmão caçula Nicolas Fernandes, por todo o amor que eu tenho e recebo, no qual me incentiva diariamente a correr atrás dos meus objetivos.

À minha tia Vanessa Espínola, que se mostrou uma das maiores apoiadoras do meu percurso como pessoa, com o seu coração imenso, sempre esteve ao meu lado e se fez presente nos meus melhores e piores momentos da minha vida.

À minha tia Wirla Pontes, que foi a precursora desse projeto, idealizou junto a mim, o projeto que hoje é apresentado, sempre estando presente me apoiando e incentivando por todo esse trajeto, hoje posso dizer que além de uma tia incrível, inspiradora, também é uma amiga e colega de projeto.

Aos meus avós Joaquim Casado e Geni Pontes, que desde criança me deram muito amor e carinho, me deram o apoio em muitos momentos da minha vida e torcem pelo meu sucesso, os amo muito.

Ao meu namorado Matheus Firmino por ter sido meu maior companheiro desse trajeto, me incentivando, aconselhando e sendo meu porto seguro, sempre estando ao meu lado nos meus altos e baixos da vida. Agradecer também à sua família que me acolheu e me apoiou por diversas vezes, serei eternamente grata.

Aos meus amigos Hélio Soares e Gabriel Jales, meus grandes amigos, colegas, companheiros de projeto, que me guiaram e me ajudaram a estar hoje aqui, sem vocês eu não estaria onde estou hoje.

Também agradecer as minhas amizades de vida Beatriz Maia, Marcelo Rossi, Layla Delgado, Lara Ventura, Jonas Rodrigues, André Luiz, Larissa Marques, Leticia Jacome, a "Família ML" e a todos os meus amigos que não estão mencionados mas têm a minha amizade e sabem disso.

Agradeço principalmente à UERN, por ter me proporcionado um ensino de qualidade, gratuito, humanizado e por ser uma universidade acolhedora, onde tive a oportunidade de crescer como pessoa e profissional.

Ao meu orientador Dr. Maximiliano Lopes, que me ajudou, incentivou, se empolgou junto a mim e me guiou durante todo o meu projeto, serei também eternamente grata.

Ao meu coorientador e amigo Exlley Santos, quem foi essencial para conseguir concluir minha monografia, por ter aceitado entrar nessa jornada e por me apoiar, me guiando até conseguir alcançar o meu objetivo final.

Aos meus professores que estiveram durante toda a minha trajetória dentro da universidade, onde tive a oportunidade de absorver os conhecimentos prestados por todos, como também pude aprender muito diante de todos os meus erros e acertos dentro do curso.

Agradeço ao PET Ciência da Computação e ao tutor Dr. Rommel Lima, por me possibilitar a oportunidade de conhecer pessoas incríveis dentro do programa, por ter passado minha universidade quase que por completa dentro do mesmo, tendo vivido momentos inesquecíveis e enriquecedores e pelo professor que apoiou os projetos que tive a oportunidade de participar e me orientado durante todo o meu percurso.

À todos o meu grande obrigado.

“Pense na beleza da vida. Observe as estrelas e
veja-se correndo com elas.”

Marco Aurélio.

RESUMO

O laboratório Muotri, localizado na Universidade da Califórnia, em San Diego, trabalha na pesquisa sobre a organogênese de um cérebro humano e outros estudos relacionados. Neste laboratório, são armazenados dados de diversas células catalogadas para serem feitos estudos na área. No entanto, a forma e o local que estes dados estão armazenados não contribui para organização e facilidade na manutenção e análise dos dados. Com base nesta problemática, este documento tem como objetivo detalhar e desenvolver uma análise de arquitetura orientada a objetos do sistema e definir um banco de dados que contribui na organização e análise das informações que são trabalhadas dentro do laboratório. Assim, essa análise foi desenvolvida com base em alguns conceitos e boas práticas levadas em consideração na engenharia de software e banco de dados, utilizando diagramas UML para representar e documentar o sistema. Por fim, foi elaborado um banco de dados com as ferramentas PostgreSQL, Node.js, Prisma e Insomnia com as tabelas necessárias para armazenar dados das entidades que são utilizadas pelo laboratório baseado na análise feita durante o projeto.

Palavras-chave: Análise de Sistemas; Banco de dados; Muotri Lab; Arquitetura Orientada a Objetos;

ABSTRACT

The Muotri laboratory, located at the University of California, San Diego, works on research into the organogenesis of a human brain and other related studies. In this laboratory, data from various cataloged cells are stored for studies in the area. However, the way and place that this data is stored does not contribute to the organization and ease of maintenance and analysis of the data. Based on this problem, this paper aims to detail and develop an object oriented architecture analysis of the system and define a database that contributes to the organization and analysis of the information that is worked on inside the laboratory. Thus, this analysis was developed based on some concepts and good practices taken into consideration in software and database engineering, using UML diagrams to represent and document the system. Finally, a database was created with the tools PostgreSQL, Node.js, Prisma and Insomnia with the necessary tables to store data from the entities that are used by the laboratory based on the analysis done during the project.

Keywords: System Analysis; Database; Muotri Lab; Object Oriented Architecture;

LISTA DE SIGLAS

LIMS	Laboratory Information Management System
ASCLS	The American Society for Clinical Laboratory Science
UCSD	University of California San Diego
OO	Object Oriented
UML	Unified Modeling Language
API	Application Programming Interface
ORM	Object-Relational Mapping
OMG	Object Management Group
HTTP	Hypertext Transfer Protocol
GUI	Graphical User Interface
JSON	JavaScript Object Notation

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

As tendências mundiais na área da tecnologia se especializam a cada ano em que se passa. São criadas novas gerações de computadores, de aplicações, automatizações de processos, dentre diversas outras novidades voltadas à evolução tecnológica. No início no século XX, é visto um amadurecimento das instituições em relação ao desempenho de projetos na tecnologia, com isso, é visto que atualmente a função da tecnologia, juntamente com a da informação/conhecimento, se inter-relacionam com outras funções dentro do ambiente organizacional para gerar um melhor desempenho dentro de qualquer área correlata. (SILVA, 2003)

Quando se lida com a área da saúde, a mesma, em conjunto com a tecnologia, trazem um grande impacto em suas soluções, porém a mesma enfrenta uma dificuldade quando se trata de aliar a parte humanitária com o avanço tecnológico da área, nela é contida uma grande influência humana no desempenho das funções em suas atividades, seja atuando diretamente com outras pessoas, seja em formato de pesquisas laboratoriais ou em gestão do âmbito de pesquisa

Neste projeto nós respeitamos o código de ética dos profissionais que estão dentro do laboratório, segundo a ASCLS - The American Society for Clinical Laboratory Science, os mesmos devem manter a reputação de seu espaço com honestidade, integridade, competência e confiabilidade. Como há um manejo de dados sensíveis, é sabido que o impacto pode ser positivo se aplicada de forma correta, quanto o contrário, logo há uma grande necessidade de cuidado no desenvolvimento diante do citado.

É sabido também que as pessoas têm dificuldade em adaptar-se à grandes mudanças, porém da mesma forma que a tecnologia vem para revolucionar, ela também está entre nós para nos auxiliar diante das nossas maiores necessidades.

Segundo Thimbleby (2013, p.161):

> "A natureza humana não muda, pelo menos não nas escalas de tempo da tecnologia. As estruturas de autoridade na área da saúde, a divisão do trabalho, a pretensão de que os médicos sabem tudo e outros fatores humanos demoram a mudar. Apesar do nosso conhecimento da teoria dos germes e antissepsia, ainda somos resistentes a lavar as mãos."

Como ocorrem avanços tecnológicos a todo momento, é necessário tempo para inserir resoluções tecnológicas dentro de um ambiente novo onde o sistema de organização ainda se

encontra desalinhado da atualidade, nisso, é importante implementar certos componentes junto às pessoas que irão utilizar o mesmo, sempre respeitando a estrutura local do trabalho, com suas hierarquias e seu fator humano.

O Muotri Lab é um laboratório de pesquisa na Universidade da Califórnia em San Diego, que atualmente entrega pesquisas sobre a organogênese de um cérebro humano, tratando sobre a complexidade do órgão para explicar o que se diferencia dos demais primatas. Esses estudos diferenciam as células-tronco, recriando “organoides cerebrais” em um ambiente controlado de um laboratório.

Liderado pelo Dr. Alysson Muotri, o mesmo tem trabalhado em implicação de modelos de doenças humanas, determinando os mecanismos moleculares e celulares que conduzem a distúrbios neurológicos complexos, como o autismo. Também dentro dos laboratórios são desenvolvidas novas abordagens de tratamento para essas doenças, como novos medicamentos e abordagens terapêuticas. (Muotri Lab, 2022)

1.1 Motivação

A ferramenta atual da gestão do laboratório contém limitações que ocasionam na dificuldade de organizar o ambiente. O gestor detém o controle das tabelas nas quais mostram os freezers, onde é armazenada a capacidade de cada um, quais estão ocupados e quais células estão armazenadas dentro das caixas.

Fora isso, o gestor necessita do andamento do processo da pesquisa, atualizações sobre, quais pesquisadores estão com projetos ativos e quais células estão sendo retiradas para pesquisa.

A atualização precisa ser feita diariamente para conseguir monitorar as ações, porém, muitas informações acabam se perdendo devido a não existir um local onde ocorra essa interação instantânea, o pesquisador pode não enviar à tempo certas informações, onde acaba confundindo o gestor quando ele precisa visualizar estas informações.

Então, quando falta algum recurso dentro do laboratório, é preciso visualizar se realmente há a necessidade de compra, se já existem células de certo tipo dentro de uma caixa e que não esteja sendo usada, ou se é uma célula que precisaria ser retornada, se ela foi testada apropriadamente, entre outros problemas causados por essa falta de informação sendo entregue em tempo hábil.

O ambiente possui uma grande demanda de produtos e necessita de atualizações onde o gestor consiga analisar as atividades para gerar atualizações sobre os itens administrados por cada projeto, como freezers, caixas de projetos, frascos, itens do laboratório de suporte para pesquisas.

Além disso, existem projetos atualizados por pesquisadores contratados pela universidade, os quais podem tanto criar um projeto, como também ser um suporte do mesmo, eles manuseiam as células disponibilizadas e fazem estudos com as mesmas, elas podem ser tanto guardadas, como clonadas para serem guardadas nas caixas do projeto, dentro desse projeto, existem itens utilizados pelos pesquisadores, o produto mais utilizado pelos mesmos são as células, armazenadas em freezers, os quais têm acesso restrito ao laboratório e apenas é aberto para os pesquisadores utilizarem dentro de suas pesquisas.

Um dos equipamentos com maior necessidade de gerenciamento, como citado anteriormente, é o freezer, separados entre os freezers compartilhados e os que armazenam conteúdos exclusivos de cada projeto. Dentro de cada freezer, existem torres de armazenamento, onde cada torre consegue armazenar uma quantidade de caixas e dentro dessas caixas, existem frascos que armazenam células para pesquisa e cada uma possui um controle de qualidade comprovada antes de ocorrer o deslocamento para um projeto ou local novo. Pode existir uma célula clone dela armazenada também para estudo na caixa de um projeto do laboratório.

Desta forma, há uma necessidade de desenvolver um sistema pró-ativo que indica processos que ocorrem dentro do laboratório e alertas sobre as atividades dos pesquisadores e gestores, atualizando todas as ações executadas. Devido à alta proporção de dados onde necessitam ser armazenados, é necessário desenvolver uma arquitetura de dados que possibilite a visualização dos processos que acontecem dentro do ambiente.

1.2 Objetivos

Levando em consideração tudo que foi abordado, é de objetivo deste projeto contribuir com a gestão de todo o ambiente laboratorial do Muotri Lab, auxiliando na organização dos equipamentos utilizados, onde foram priorizados os freezers e a visualização deles, já que possuem dentro dos mesmos material biológico de grande impacto no âmbito da saúde.

1.3 Estrutura do Trabalho

Dentro do contexto citado anteriormente, o trabalho proposto tem como objetivo contribuir com a gestão do laboratório Muotri através da análise do sistema e implementação de um banco de dados para o gerenciamento acerca dos relatos de dificuldade em organizar e manusear dados.

Tendo como o intuito atingir o objetivo geral, podem ser listados alguns objetivos especificados, como:

- Na seção 2 é apresentado os principais assuntos relacionados com este trabalho, como os conceitos de arquitetura orientada a objetos, banco de dados e definição do modelo UML e seus métodos que foram usados para definir a solução do problema.
- Na seção 3 é definida a proposta de resolução do trabalho, contendo os estudos de caso, como toda a arquitetura do projeto, a api e as ferramentas utilizadas para seu desenvolvimento
- Na seção 4 é exposta as conclusões do projeto e os trabalhos futuros; no Apêndice, todos os requisitos do sistema e casos de uso desenvolvidos.

2 REFERENCIAL TEÓRICO

Nesta seção é abordado sobre a Arquitetura Orientada a Objetos e sua contextualização, discutindo sobre seus conceitos, modelagem UML, como também sobre o banco de dados orientado a objetos, seus mecanismos e padrões a se usar, além do estado da arte atual do tema.

2.1 Arquitetura Orientada a Objetos

O primeiro estágio de qualquer processo de projeto de software é o desenvolvimento de uma compreensão dos relacionamentos entre o software que está sendo projetado e seu ambiente externo. Isso é essencial para decidir como oferecer a funcionalidade requerida para o sistema e como estruturar o sistema para se comunicar com seu ambiente. (Sommerville, 2011, pg.126)

De acordo com Mendes (2017, pg.50-51), para a elaboração de um sistema, é necessária a organização das funcionalidades do mesmo, seguindo fluxos prioritários para um projeto ser criado. Existem três fases genéricas que compõem a sua criação: Definição, Desenvolvimento e Manutenção, demonstradas no quadro 1.

Quadro 1 - Fases de Criação de um Projeto

Definição	Trata da identificação das informações que devem ser processadas, suas funções e o desempenho esperado, além disso, define a interface a ser utilizada, as tarefas que o sistema irá suportar, definir o perfil dos usuários, dentre outros critérios.
Desenvolvimento	É a fase que, depois de definidos os requisitos do sistema, ocorre a estruturação dos dados e a arquitetura do software, também convertendo para uma linguagem de programação, onde realizam os testes e os avaliam.
Manutenção	É a parte onde se consideram modificações e/ou correções necessárias no sistema para atender aos requisitos do usuário.

Fonte - (Autoria Própria)

No processo de desenvolvimento existem dois tópicos principais de interesse que se trata da implementação das funcionalidades do sistema e a atividade anterior na qual orienta

essa criação e é justamente o projeto da arquitetura do software, onde trazem essas informações para as decisões arquitetônicas.

A Arquitetura Orientada a Objetos auxilia os desenvolvedores em diversos fatores, tanto em entender a ideia de negócio, estruturar seus objetos e entregar atividades que conseguem simular o dia a dia em formato de programa (Rudolf Pecinovský, 2013).

Logo, de acordo com Weisfeld (2009), o comportamento do objeto se dá pelo que ele pode fazer, quando são usadas linguagens procedurais, seus comportamentos são definidos por procedimentos, funções e subrotinas. Na programação orientada a objetos, esses comportamentos são contidos em métodos, onde se consegue invocá-los quando necessário, enviando uma mensagem para a ação.

Uma vantagem da programação orientada a objetos se dá pelo fato dos dados manipulados pelas operações são ambos encapsulados dentro de um objeto, fazendo com que quando um objeto for invocado, ele irá entregar todos os dados e comportamentos que estão dentro dele. Então um programa que usa uma tecnologia OO traz metodologias para desenvolvimento relacionados aos objetos, seus componentes serão citados logo abaixo.

2.1.1 Classes

Na linguagem orientada a objetos a classe vem antes de todos os componentes de um projeto. Para melhor explicar, quando se é lidado com uma tabela de banco de dados, a definição de uma tabela por si só é reconhecido como uma classe e os objetos seriam as linhas de uma tabela (dados). Em resumo, um objeto não pode ser instanciado sem uma classe.

2.1.2 Atributos

Os dados armazenados em um objeto representam o seu estado, e na terminologia da programação orientada a objetos, esses dados são chamados de atributos. Esses atributos dentro da programação são aqueles que contém as informações necessárias para armazenar os dados importantes que diferenciam cada objeto.

2.1.3 Métodos

Os métodos implementam os comportamentos que uma classe irá ter, em cada objeto instanciado por uma classe tem métodos definidos pela mesma. O método deve implementar os comportamentos que são chamados por outros objetos (mensagens) ou proporcionar o

comportamento interno e fundamental para uma classe funcionar. Seus comportamentos internos são métodos privados onde não são acessíveis por outros objetos.

2.1.4 Mensagem

As mensagens são os mecanismos entre os objetos, os métodos por exemplo de um objeto quando invocam outros métodos, eles enviam uma mensagem para comunicação e apenas métodos públicos podem ser invocados por outros objetos.

2.1.5 Encapsulamento

Uma das maiores vantagens de usar objetos para guardar dados de uma classe é por não precisar revelar todos os seus atributos e comportamentos. Em um bom design OO, o objeto deve apenas revelar as interfaces onde outros objetos têm de interagir com ele. Logo, detalhes não pertinentes para o uso do objeto devem ser ocultados de todos os objetos. O Encapsulamento é definido pelo fato de que objetos contém atributos e comportamentos.

2.1.6 Encapsulamento de Dados

O encapsulamento de dados permite que o desenvolvedor do ambiente consiga pensar no nível da estrutura de dados e suas operações desempenhadas para depois conseguir detalhar a estrutura e operações dos dados a serem implementados. (Schach, 2008, pg.192)

2.1.7 Interfaces

A interface define os significados fundamentais da comunicação entre os objetos. Cada design de classe especifica as interfaces para a operação e instanciação própria dos objetos. Qualquer comportamento que o objeto fornece deve ser invocado por uma mensagem enviada, essa mensagem usa a interface para comunicação. Na maioria das linguagens OO, os métodos que são parte da interface são elegidos como públicos. (Weisfeld, 2009, pg. 19)

2.2 Modelo UML

Este tópico foi baseado na documentação da OMG (2017), que é uma organização internacional onde detalha os padrões da indústria de computadores e tem a UML como um de seus recursos.

A especificação que define a Linguagem de Modelagem Unificada (UML) tem como objetivo prover ferramentas para análise, design e implementação de sistemas baseados em software, como também para modelagem de negócios e processos similares. Essas

ferramentas auxiliam os arquitetos de sistemas, engenheiros de software e desenvolvedores de software no desenvolvimento de um projeto orientado a objetos. (OMG, 2017)

As versões iniciais do UML (UML 1) foi originada com três métodos orientado à objetos (Booch, OMT e OOSE), onde durante sua concepção foram incorporadas as melhores práticas do design de linguagem de modelagem, da programação orientada à objetos e das linguagens de descrição arquitetural. Em relação à UML 1, a revisão atual foi aprimorada tal que suas regras de sintaxe e semântica abstratas se tornaram mais precisas, sua estrutura de linguagem se tornou mais modular e contém uma capacidade aprimorada para modelar sistemas de grande escalabilidade.

Um dos principais objetivos da UML é avançar o estado da indústria, permitindo a interoperabilidade da ferramenta de modelagem visual de objetos. No entanto, para permitir uma troca significativa de informações de modelo entre ferramentas, é necessário um acordo sobre semântica e sintaxe. A UML atende aos seguintes requisitos:

- Definição formal baseado em um metamodelo no qual especifica a sintaxe abstrata da UML. A sintaxe abstrata define o conjunto de conceitos de modelagem UML, seus atributos e seus relacionamentos, bem como as regras para combinar esses conceitos para construir modelos UML parciais ou completas.
- Uma explicação detalhada da semântica de cada conceito de modelagem UML. A semântica define, de forma independente da tecnologia, como os conceitos UML devem ser realizados pelos computadores.
- Uma especificação do ser humano elementos de notação legível para representar os conceitos individuais de modelagem UML, bem como regras para combiná-los em uma variedade de diferentes tipos de diagramas correspondentes a diferentes aspectos dos sistemas modelados.

A sintaxe abstrata da UML é especificada usando um modelo UML chamado metamodelo UML. Este metamodelo usa construtores de um subconjunto restrito de UML e é usado para construir metamodelos.

Na modelagem UML, ela sempre irá ser desenvolvida baseada em um modelo de algo, esse projeto que possa estar sendo modelado pode ser genericamente considerado um sistema de algum domínio do discurso. O modelo então faz algumas declarações de interesse sobre

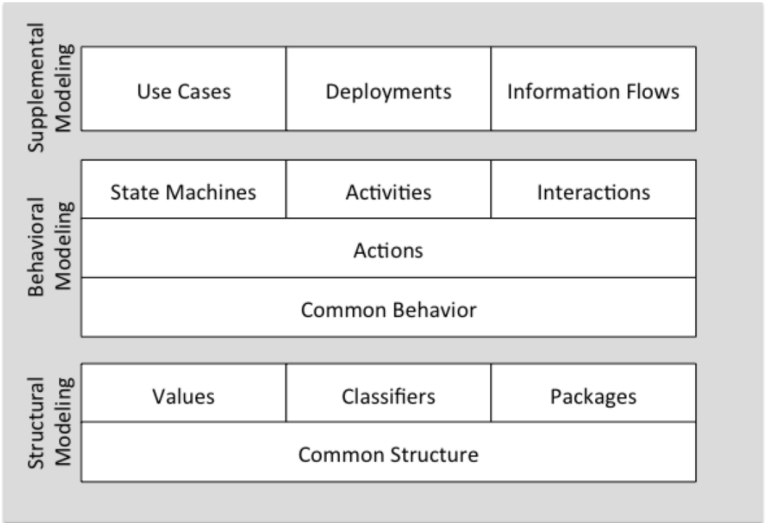
esse sistema, abstraindo todos os detalhes dele que poderiam ser descritos, de um determinado ponto de vista e para um determinado propósito.

Para um sistema existente, o modelo pode representar uma análise das propriedades e comportamento do sistema. Já para um sistema planejado, o modelo pode representar uma especificação de como o sistema deve ser construído e se comportar. Um modelo UML consiste em três categorias principais de elementos de modelo, onde cada um pode ser usado para fazer declarações sobre diferentes tipos de objetos individuais dentro do sistema que está sendo modelado. Essas categorias são:

- **Classificadores**: O classificador descreve um conjunto de objetos. Onde o objeto é um indivíduo com um estado e relacionamentos com outros objetos. O estado de um objeto identifica os valores para esse objeto de propriedades do classificador do objeto.,
- **Eventos**: Ele descreve um conjunto de ocorrências possíveis. A ocorrência é algo que acontece e tem alguma consequência em relação ao sistema.
- **Comportamentos**: Nele descreve um conjunto de execuções possíveis. Uma execução é um conjunto de ações durante um período de tempo que podem gerar e responder a ocorrências de eventos, incluindo acessar e alterar o estado dos objetos.

Na figura 1 é apresentada uma demonstração mais detalhada das áreas semânticas da UML dentro de suas respectivas categorias com uma noção maior do que está dentro de cada área.

Figura 1 - Áreas Semânticas da UML



Fonte - (OMG UML, 2017)

Para modelagem do sistema, é importante utilizar o padrão UML para criação dos elementos que estão dentro da arquitetura, para isso, são desenvolvidos diagramas UML que auxiliam nesta parte, tópico que será abordado logo a seguir.

2.2.1 Diagrama UML

A UML foi criada com o objetivo de estabelecer uma linguagem de modelagem visual comum, sendo uma ela uma linguagem sintaticamente e semanticamente rica, para arquitetura, design e implementação de sistemas, tanto do ponto de vista de comportamento, quanto estruturalmente. Além disso, possui diferentes tipos de diagramas com o intuito de descrever o limite, a estrutura e o comportamento do sistema e os seus objetos. (Lucidchart, 2022)

Segundo a OMG (2005), a modelagem de um projeto de software antecede o desenvolvimento com código, sendo uma parte essencial para grandes projetos e útil para outros de médio e pequeno porte, esse tipo de modelo tem um desempenho paralelo ao da programação, auxiliando assim em qualquer implementação que exista posteriormente.

No sentido de precificação, um projeto acaba se tornando muito mais custoso quando nele existe pouco planejamento e muita implementação, já que para modificar uma alteração, é necessário de muito mais tempo e recursos.

Usando um modelo UML, os responsáveis pelo sucesso de um projeto de desenvolvimento de software podem se assegurar de que a funcionalidade do negócio está completa e correta, as necessidades do usuário final são atendidas e o design do programa suporta os requisitos de escalabilidade, robustez, segurança, extensibilidade e outras características necessárias para um bom desempenho.

2.2.2 Diagrama de Caso de Uso

O diagrama de caso de uso modela o comportamento do sistema e ajuda a capturar os requisitos do sistema. Ele descreve as funcionalidades de forma direta e o escopo de um sistema. Além disso, identifica as suas interações entre suas funcionalidades e seus atores, sem a preocupação de mostrar como o sistema funciona internamente. Serve tanto para ilustrar e definir os requisitos do sistema como um todo e dos seus principais componentes. (IBM, 2021)

Na criação de um diagrama de caso de uso, existem passos que ajudam a definir o contexto e a modelagem dentro dos padrões UML. No qual há estágios de criação

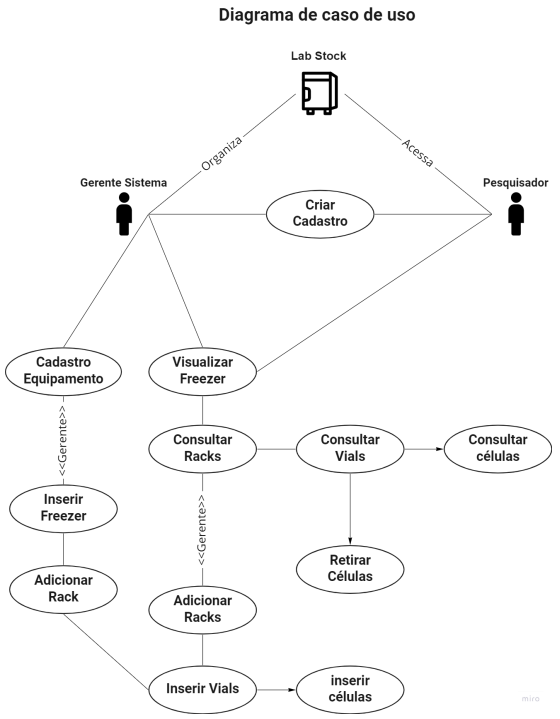
recomendados para ter um diagrama útil e implementável em um produto. Seus principais passos para atingir o resultado esperado estão em:

- Definir os atores existentes dentro do sistema, eles representam o papel de um usuário no qual interage com o sistema que está sendo modelado, como um usuário humano, uma organização, equipamentos, dentre outros. Nessa etapa existem diversos formatos de pesquisas e entrevistas que podem auxiliar a equipe desenvolvedora na definição do(s) ator(es) do sistema.
- Gerar casos de uso que descrevem a função do sistema em relação ao usuário. Compreendendo os atores envolvidos, os casos de uso serão capazes de conter informações detalhadas sobre o sistema, sobre os usuários do sistema, relacionamentos entre o sistema e os usuários e o comportamento necessário do sistema.

Tendo assim definido esses dois passos, a modelagem do diagrama se torna mais simples e fácil de resolver, pois suas conexões já estarão bem definidas no escopo do projeto.

Na figura 2, é mostrado como um diagrama de caso de uso se organiza, utilizando o modelo do sistema desenvolvido de forma simplificada.

Figura 2 - Exemplo Caso de Uso Muotri Lab



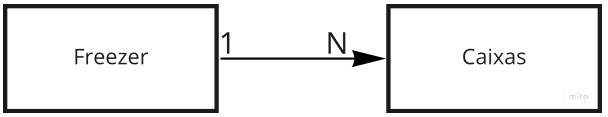
Fonte - (Autoria Própria)

2.2.3 Diagrama de Classes

O diagrama de classes UML é uma notação gráfica usada para construir e visualizar sistemas orientados a objetos. Um diagrama de classes UML é um tipo de diagrama de estrutura estática no qual descreve um sistema mostrando alguns atributos, como as classes, os seus atributos, os métodos e mostra como são as relações entre os objetos.

Os diagramas de classe em UML podem ser expressos em diferentes níveis de detalhamento. No desenvolvimento de um modelo, o principal objetivo é identificar os objetos essenciais para representá-los como classes. A maneira inicial mais simples para conseguir demonstrar é colocar o nome da classe em uma caixa, sendo assim conseguindo observar a existência de uma associação entre essas classes. (Sommerville, 2011, pg.90) Um exemplo simples que pode ser dado é na questão dos freezers, onde eles podem ter uma relação direta, essa relação visualmente pode ser descrita de forma simples, como representado na figura 3.

Figura 3 - Diagrama de Classes Simples



Fonte - (Autoria Própria)

Conforme o nível de detalhamento de uma classe, é possível chegar a um modelo semântico de banco de dados, onde mostram as entidades dos dados, seus atributos associados e as relações entre essas entidades Dentro do detalhamento das informações da classe, pode inserir seus atributos e tipos e para chegar nesses componentes existem duas técnicas utilizadas, que são a Generalização e a Agregação.

Generalização: Consiste em generalizar uma classe para aprender as suas características, geralmente auxilia no agrupamento de informações comuns em um único lugar, na programação orientada a objetos a generalização é implementada usando a herança de classes.

Agregação: A agregação une partes diferentes em um mesmo objeto, ou seja, se existem dois objetos que estejam associados diretamente a um objeto só, pode existir a agregação entre eles.

2.3 Trabalhos Relacionados

Nesta seção, encontram-se propostas de software com as soluções mais próximas ao projeto, todos os mencionados já estão disponíveis no mercado e possuem um nível de maturidade do sistema alto. Os comparativos estão enviesados para o sistema de gerenciamento do ambiente laboratorial e seus recursos.

Quadro 2 - Trabalho relacionado (Lab Collector)

Nome	Lab Collector
Desenvolvedor	AgileBio
Plataforma	All-in-One Lab notebook
Descrição do Software	O aplicativo fornece atividades diversas e customizáveis para que o gerente do espaço possa organizar da melhor forma, como triagem, detecção, produção, análise, qualquer que seja o domínio de aplicação, além de estabelecer um Sistema de Gestão de Qualidade (QSM).

Fonte - AgileBio, 2022

Quadro 3 - Trabalho relacionado (VIDA - Sistema para Laboratório)

Nome	VIDA - Sistema para Laboratório
Desenvolvedor	TechMobil
Plataforma	Web
Descrição do Software	Voltado ao Gerenciamento de laboratórios de análises clínicas, ele integra toda a gestão do ambiente, desde o registro de um paciente na recepção, até a movimentação financeira do lugar, gerindo a coleta, estoque, soroteca, gestor e o faturamento eletrônico.

Fonte - TechMobil, 2022

O Lab Collector pode ser afirmado como uma solução amadurecida e ampla, tendo a possibilidade de customizar toda a experiência do sistema, o seu diferencial está em ser um sistema completo, com diversas ferramentas genéricas que abrangem diversas áreas, porém, acaba por apresentar uma proposta diferente do projeto que é desenvolvido, já que o sistema deve apresentar especialidades que acabam por não serem abrangidas pelo Lab Collector.

O VIDA é feito para laboratórios de análises clínicas, que também se mostra maduro em relação ao mercado brasileiro, seu diferencial está em conseguir lidar com diversos setores dentro de um ambiente organizacional, conseguindo gerar valor desde a parte laboratorial, até a parte de gestão de pessoas e clientes. Como o Lab Collector, ele também possui diversas ferramentas, apesar de não possuir a possibilidade de customização. Comparado ao projeto, sua semelhança está em possuir a rastreabilidade de informações sobre entradas e saídas de

informações monitoradas pelo gestor, porém acaba se tornando sua única similaridade, já que o mesmo não demonstra proximidade na proposta de solução final.

3 GBD PROPOSTA

Neste capítulo será abordado o desenvolvimento do sistema, e o levantamento de funcionalidades necessárias para o desempenho e a entrega de resultados satisfatórios.

Para chegar em uma solução viável para o negócio, foram necessárias entrevistas com o cliente para conseguir compreender as suas dificuldades, onde nos foi mostrado o laboratório, sua estrutura, seus recursos e a dinâmica ocorrida dentro dele.

Outro recurso disponibilizado, dentro dos encontros para levantamento de requisitos, foram planilhas com todos os itens que haviam dentro do espaço de pesquisa, como freezers, células e afins. Com isso, todo o sistema foi baseado nas informações entregues pelo cliente, este que será apresentado de forma detalhada nos tópicos a seguir.

3.1 Visão Geral

O Laboratório Muotri tem como fundamento principal fazer estudos na área da neurociência, nele existem pesquisadores de todo o mundo estudando células, fazendo experimentos, dentre diversas formas de análises com finalidade de obter novas descobertas científicas promissoras para a sociedade.

Em sua estrutura organizacional, onde cada indivíduo exerce um papel, os principais destaques estão em dois papéis mais importantes que foram destacados de forma clara a sua importância, sendo então àqueles que utilizam o laboratório, logo, usufruem dos recursos prestados, como também os que gerenciam esses recursos e precisam monitorar os mesmos. Para auxiliar o papel da administração existem auxiliares nos quais estão encarregados de ajudar a gerenciar o espaço laboratorial.

Com isso, os três principais usuários definidos dentro dessa problemática são o Pesquisador, Administrador e Gerente. Para ficar melhor exemplificado:

- Pesquisador: O usuário pesquisador é aquele que irá retirar os recursos dentro do equipamento que armazena-os e utiliza-os. Para identificar o pesquisador, é armazenado o seu nome, endereço de email, identificador para login e a senha.

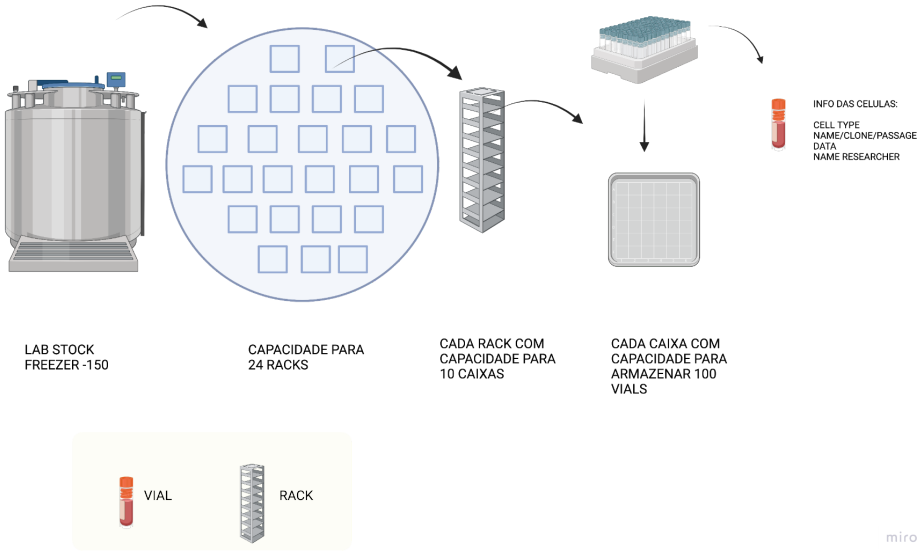
- Administrador: É o usuário principal, ele faz a parte organizacional dos equipamentos e dos recursos, monitorando também as ações de todos os usuários dentro do sistema e garantindo um controle de qualidade dentro do ambiente, é armazenado o seu nome, endereço de email, identificador para login e senha.
- Gerente: É o usuário auxiliar do administrador, ajudando no controle dos equipamentos e do usuário Pesquisador, é armazenado o seu nome, endereço de email, identificador para login e senha.

Estes usuários se diferem de acordo com a hierarquia das atribuições de cada serviço, computacionalmente falando, os três mantêm identificadores com valores diferentes em cada login.

Diante do que foi abordado, inicialmente, o maior objetivo foi identificar os equipamentos e itens primordiais com maior necessidade de gerenciamento e organização. Dentre eles estão: Freezer, Racks, Caixas, Frascos e Células. Contextualizando, existem freezers a -150° voltados ao armazenamento de células para estudo.

Na organização do freezer, existem racks com um espaço para guardar caixas e dentro dessas caixas existem frascos (vials) que guardam as células. A figura 4 mostra de forma visual como são os freezers:

Figura 4 - Exemplo dos equipamentos



Fonte - (Autoria Própria)

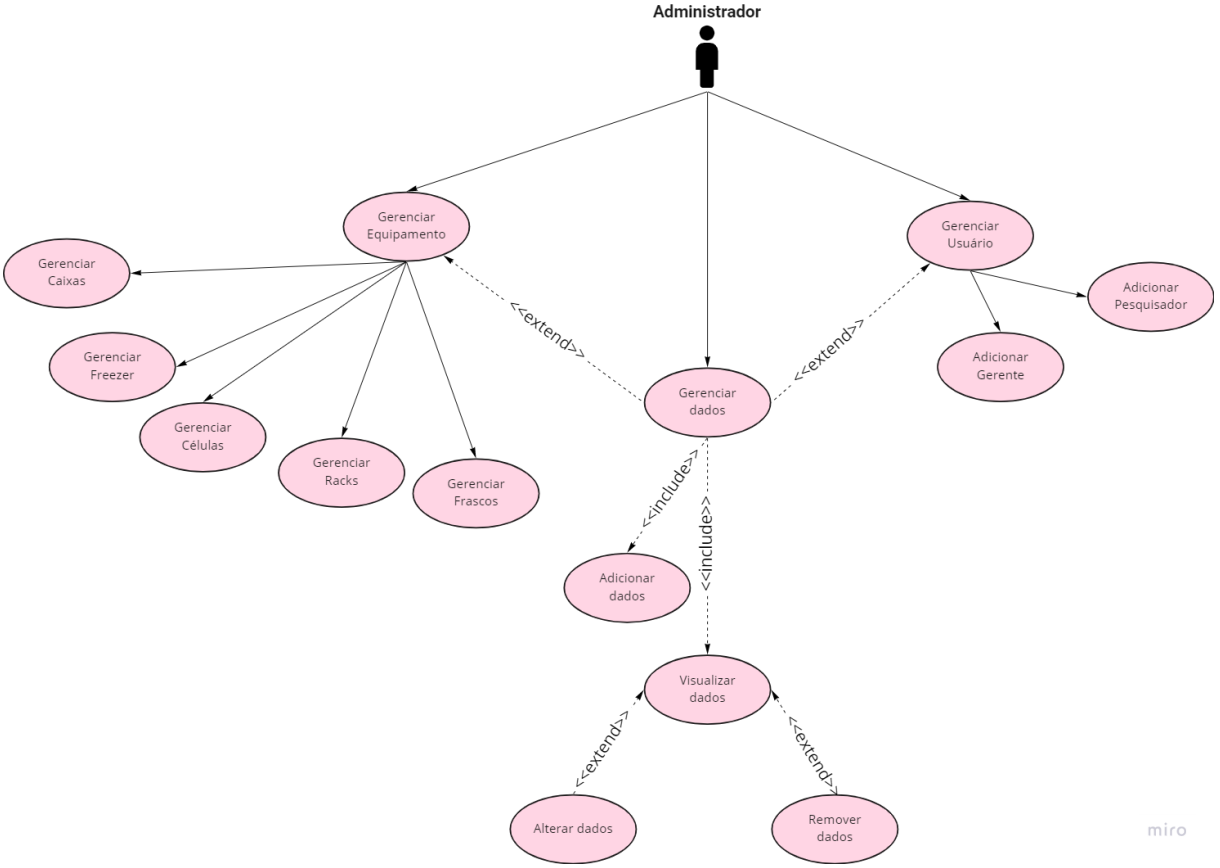
Com base nas informações explicadas até o momento, foi desenvolvido um diagrama de caso de uso para definir as ações exercidas por cada agente e suas relações com os itens do sistema. Nos tópicos a seguir serão explanados os elementos do sistema organizados, baseados em UML e suas ferramentas de modelagem de uso.

3.2 Diagrama de Caso de Uso

Um caso de uso é um modelo de interação entre usuários de um produto de software (atores) e o próprio produto de software. Mais precisamente, um ator é um usuário desempenhando um papel específico. Um diagrama de casos de uso é um conjunto de casos de uso. (Schach, 2008, p.488)

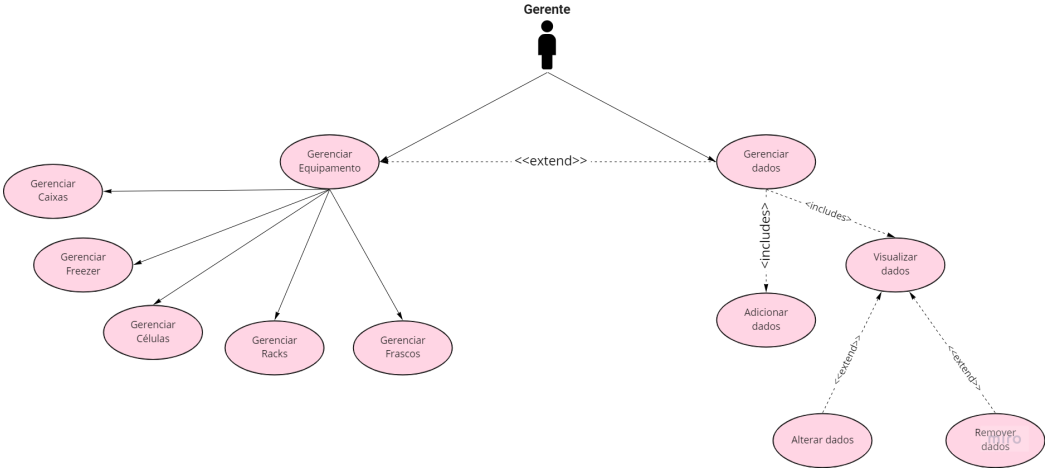
Nos diagramas demonstrados nas figuras 5, 6 e 7 são explanados os atores, ações e seus relacionamentos.

Figura 5 - Diagrama de caso de uso do Administrador



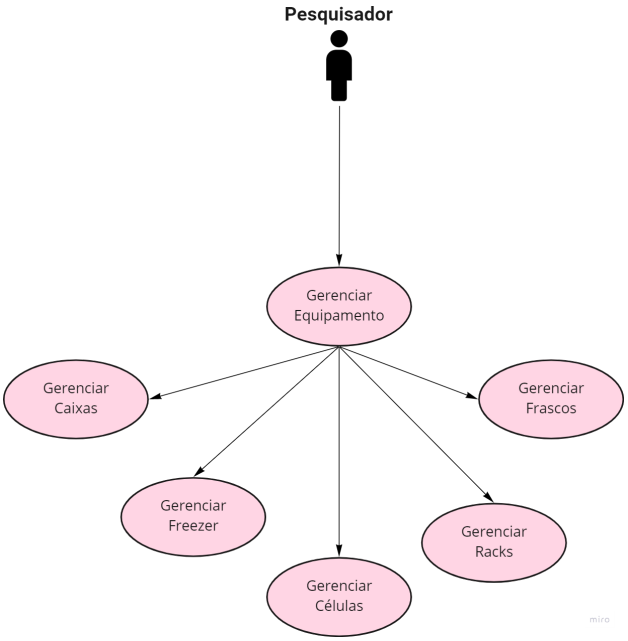
Fonte - (Autoria Própria)

Figura 6 - Diagrama de caso de uso do Gerente



Fonte - (Autoria Própria)

Figura 7 - Diagrama de caso de uso do Pesquisador



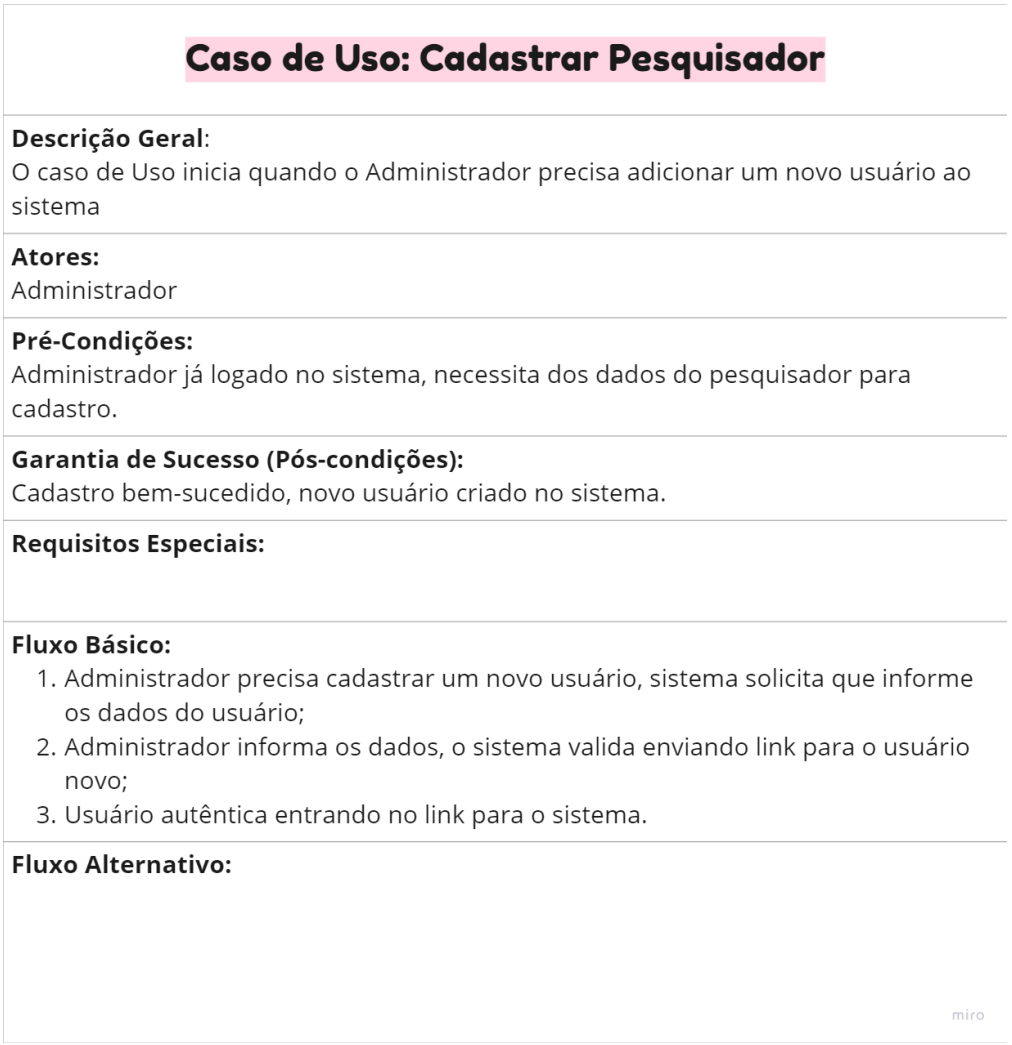
Fonte - (Autoria Própria)

3.3 Casos de Uso

Um caso de uso identifica os atores envolvidos em uma interação e dá nome ao tipo de interação. Essa é, então, suplementada por informações adicionais que descrevem a interação com o sistema. (Sommerville, 2011, pág.74)

O caso de uso demonstrado na figura 8 é referente ao cadastro dos pesquisadores dentro do sistema pelo usuário Administrador, os demais casos de uso estão dispostos dentro do apêndice deste trabalho.

Figura 8 - Caso de uso de cadastro do Pesquisador.



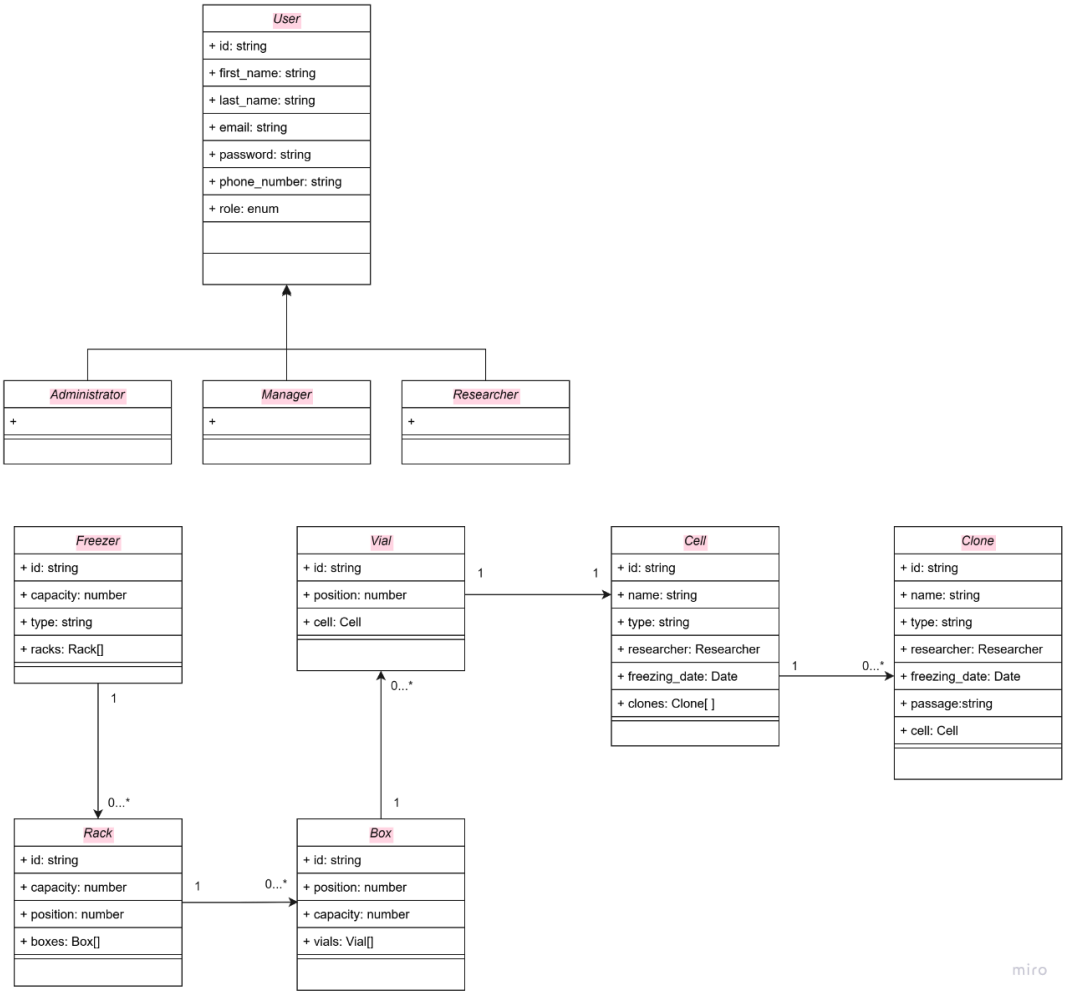
Fonte - (Autoria Própria)

3.4 Diagrama de Classes

Na engenharia de software, um diagrama de classes na Unified Modeling Language (UML) é um tipo de diagrama de estrutura estática que descreve a estrutura de um sistema mostrando as classes do sistema, seus atributos, operações (ou métodos) e os relacionamentos entre objetos. (Visual Paradigm, 2022)

O Diagrama de Classes vai servir para modelagem do banco de dados, onde ilustra as classes que foram criadas e suas relações, sendo então demonstrado na figura 9, baseado segundo o estudo do levantamento de requisitos do sistema.

Figura 9 - Diagrama de Classes



Fonte - (Autoria Própria)

3.5 Validação da Regra de Negócio

Nesta seção é explanado o processo de validação do projeto, explicando as ferramentas para validação, suas funções, como foi projetado e processo de implementação.

A validação é feita para assegurar que a regra de negócio atenda as demandas do cliente, entregando uma proposta de solução de acordo com as necessidades iniciais apontadas. De acordo com essas dificuldades nas quais foram analisadas, elas se mostram majoritariamente relacionadas ao asseguramento e observação dos dados do laboratório.

3.5.1 Ferramentas Aplicadas

Nesta seção apresenta as ferramentas usadas para validação da regra de negócio, nas quais foram de suma importância para o desenrolamento do projeto.

3.5.1.1 Prisma

Prisma é uma ORM de código aberto, que serve para uma modelagem de dados intuitiva. Ele é composto pelas seguintes partes:

- Prisma Client: construtor de consultas gerado automaticamente e com segurança de tipo para Node.js e TypeScript
- Prisma Migrate: sistema de migração
- Prisma Studio: Utiliza uma GUI para visualizar e editar dados em seu banco de dados

O Prisma Client pode ser usado em qualquer aplicativo Node.js (versões suportadas) ou em uma aplicação back-end com TypeScript (incluindo aplicativos sem servidor e microsserviços).

Fora isso, ele também consegue fazer conexões com os principais bancos de dados utilizados pelos desenvolvedores, como Postgresql, Mysql, SQLite, dentre outros. Outro ponto positivo já citado anteriormente é o editor Prisma Studio, que foi utilizado para visualizar o Banco desenvolvido e conseguir editá-lo.

3.5.1.2 Insomnia

A ferramenta Insomnia é um aplicativo de desktop multiplataforma gratuito que auxilia a projetar e interação com APIs baseadas em HTTP. Ele tem uma interface fácil de usar com funcionalidades avançadas ajudando na autenticação, geração de código e variáveis de ambiente. Ele no projeto se faz necessário para consulta do status do banco.

3.5.1.3 Node.js

O framework Node.js foi projetado para criar aplicativos de rede escaláveis utilizando a linguagem javascript, ele é um ambiente de execução *server-side,* onde consegue suportar várias conexões simultâneas, também é indicado para criar bibliotecas e frameworks web.

3.5.1.4 Postgresql

O PostgreSQL surgiu em 1968 como parte do projeto POSTGRES da Universidade da Califórnia em Berkeley. Ele é um sistema de banco de dados relacional de código aberto que utiliza a linguagem SQL combinada com várias funcionalidades que armazenam grande carga de trabalho de dados de forma segura e escalável.

O PostgreSQL, como falado anteriormente, possui diversas funcionalidades para ajudar na administração do sistema. Sendo essas funcionalidades relacionadas com os tipos de dados, integridade de dados, concorrência, performance, confiabilidade, recuperação de desastres, segurança, extensibilidade, internacionalização, busca de texto e além de outras que são capazes de adição. (The PostgreSQL Global Development Group, 2022)

Para manusear o Postgres, a ferramenta mais conhecida para o seu gerenciamento é o PGAdmin, no qual é uma ferramenta open source, tendo disponível uma GUI usada para interagir com as sessões do banco de dados, tanto em servidores locais quanto remotos. Possibilitando a execução de qualquer tipo de administração necessária para um banco de dados Postgres.

3.6 Resultados

Nesta subseção é explicado como o projeto foi implementado para validação do banco desenvolvido, que é apresentado no tópico 3.4 deste trabalho, como também as ferramentas utilizadas para se alcançar o objetivo final do mesmo.

Para criar o ambiente, foi usado o Node.js para fazer a criação do servidor e possibilitar a visualização juntos aos seus módulos. E em conjunto com ele, a ORM Prisma para facilitar na população do banco de forma simples e ágil. Dentro da sua aplicação, é tida a conexão com um banco já existente. No quadro 4, é apresentado o formato de como se conectar com o banco PostgreSQL.

Quadro 4 - Conexão com o Banco de Dados

```
datasource db {
 provider = "postgresql"
 url      = env("DATABASE_URL")
}
```

Fonte - (Autoria Própria)

No Prisma, é possível manipular o banco de dados utilizando um esboço realizado pelo desenvolvedor, com todas as informações do modelo organizado previamente das classes e seus atributos. No quadro 5, mostra como é a estrutura para se criar uma entidade do banco pelo Prisma.

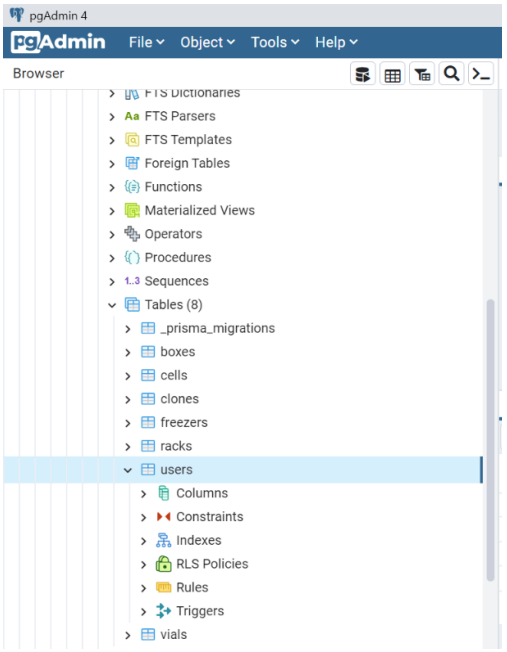
Quadro 5 - Representação da Tabela de Usuários

```
model users {
 id          String     @unique
 first_name  String
 last_name   String
 email       String     @unique
 password    String
 phone_number String?
 role        user_roles?
 created_at  DateTime?
 updated_at  DateTime?
 deleted_at  DateTime?
 cells       cells[]
 clones      clones[]
}
```

Fonte - (Autoria Própria)

Após isso, é possível constatar que todas as tabelas foram criadas no banco de dados do Postgresql utilizando o PgAdmin, mostrado na figura 10.

Figura 10 - Tabela com as Entidades



Fonte - (Autoria Própria)

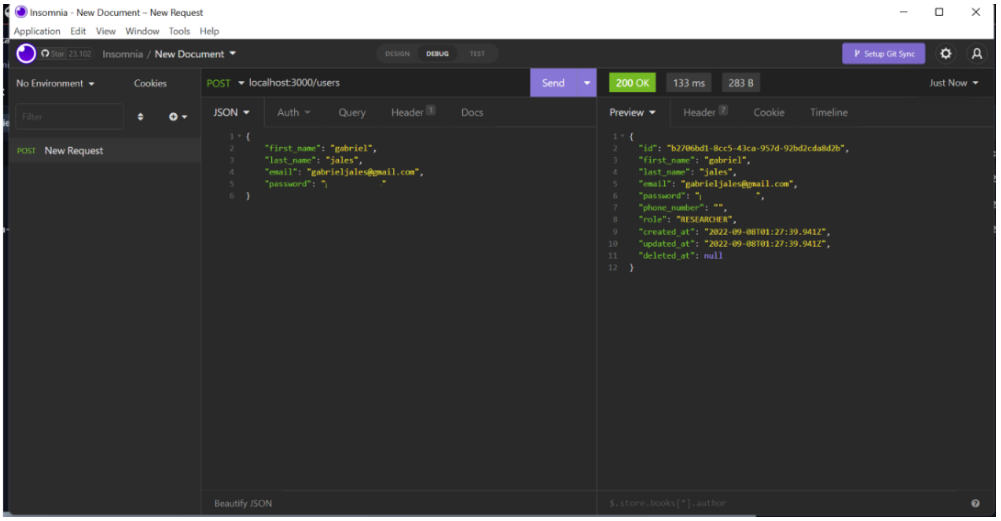
O Insomnia, irá servir para realizar requisições de uma API, onde dentro dela, é capaz de enviar informações para o banco utilizando a rota POST, por um arquivo .JSON para adicionar dados nas tabelas do banco, um exemplo de como fazer uma requisição pelo insomnia está inserido no quadro 6, e sua interface logo a seguir na figura 11.

Quadro 6 - Arquivo .JSON

```
{
  "first_name": "gabriel",
  "last_name": "jales",
  "email": "gabrieljales@gmail.com",
  "password": "**"
}
```

Fonte - (Autoria Própria)

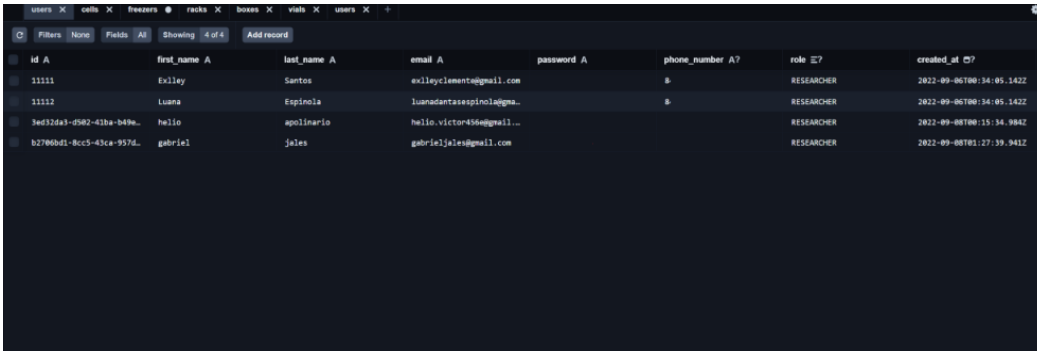
Figura 11 - Interface Insomnia



Fonte - (Autoria Própria)

Para visualizar o que foi inserido, a ferramenta Prisma Studio serviu para ter a certeza de que os dados que foram inseridos estão dentro do banco armazenados de forma exata e com os campos corretos, além de ser possível popular com dados novos pela sua interface. Na figura 12, é possível ver os dados que foram adicionados tanto pelo Insomnia quanto pelo próprio Prisma Studio.

Figura 12 - Tabela de Dados



Fonte - (Autoria Própria)

4 CONSIDERAÇÕES FINAIS

Tendo em vista que o laboratório Muotri necessitava de um recurso para gerenciamento de sua organização, foi de suma importância iniciar o processo de transição para uma plataforma capaz de armazenar seus dados eficientemente, no qual anteriormente estava sendo usado em um local que não conseguia suprir suas demandas.

Para se chegar em um sistema funcional, é fundamental documentar e avaliar todas as necessidades que um usuário tem, desde ações particulares, até interações com outras entidades dentro de um ambiente, para isso, foi essencial criar um estudo de caso capaz de identificar essas ações e modelar, para conseguir as enquadrar em um modelo computacional cabível dentro da problemática.

Considerando todo o estudo realizado para o desenvolvimento desse projeto, é entregue uma arquitetura orientada a objetos, utilizando da engenharia de software para modelar e organizar todas as informações estabelecidas dentro deste projeto, com um banco de dados funcional e dentro dos padrões de qualidade de toda organização, em que se é capaz de operar com os dados fornecidos e realizar as tarefas estabelecidas, na qual as exigências propostas pelo cliente são supridas, como armazenar dados de equipamentos e usuários, além de poder manipular e visualizá-los.

A arquitetura foi desenvolvida para facilitar a implementação de toda a estrutura restante do sistema, logo, em trabalhos futuros, há o propósito de entregar um sistema web escalonável e de acordo com a infraestrutura do laboratório.

REFERÊNCIAS

AGILEBIO. **About AgileBio**. [*S. l.*], 2022. Disponível em: https://labcollector.com/about/company/. Acesso em: 1 set. 2022.

ASCLS. **Code of Ethics**. [S. l.], 2022. Disponível em: https://ascls.org/code-of-ethics. Acesso em: 30 ago. 2022.

DATABRICKS. **ACID Transactions**. [*S. l.*], 2022. Disponível em: https://www.databricks.com/glossary/acid-transactions. Acesso em: 31 ago. 2022.

DATABASE.GUIDE. **What does ACID mean in Database Systems?**. [*S. l.*], 2016. Disponível em: https://database.guide/what-is-acid-in-databases/. Acesso em: 31 ago. 2022.

IBM. **Use-case diagrams**. [*S. l.*], 2016. Disponível em: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case. Acesso em: 30 ago. 2022.

INOVE DISTRIBUIDORA. **Software para laboratório de análises clínicas**. [*S. l.*], 2022. Disponível em: https://www.sisvida.com.br/. Acesso em: 1 set. 2022.

KONG INC. **Introduction to Insomnia**. [*S. l.*], 2021. Disponível em: https://docs.insomnia.rest/insomnia/get-started. Acesso em: 2 set. 2022.

LOSHIN, Peter. **Structured Query Language (SQL)**. [*S. l.*], 2022. Disponível em: https://www.techtarget.com/searchdatamanagement/definition/SQL. Acesso em: 28 ago. 2022.

LUCIDCHART. **O que é um diagrama UML?**: Por que usar um diagrama UML?. [*S. l.*], 2022. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-uml. Acesso em: 29 ago. 2022.

OBJECT MANAGEMENT GROUP, INC. An OMG Unified Modeling Language. **An OMG® Unified Modeling Language® Publication OMG® Unified Modeling Language® (OMG UML®)**, [*s. l.*], ed. 2.5.1, 2017. Disponível em: https://www.omg.org/spec/UML/2.5.1/PDF. Acesso em: 1 set. 2022.

OBJECT MANAGEMENT GROUP, INC. **Introduction To OMG's Unified Modeling Language**. [S. l.], 2005. Disponível em: https://www.uml.org/what-is-uml.htm. Acesso em: 29 ago. 2022.

OPENJS FOUNDATION *et al*. **About Node.js**. [*S. l.*], 2022. Disponível em: https://nodejs.org/en/about/. Acesso em: 2 set. 2022.

PECINOVSHÝ, Rudolf. OOP – Learn Object Oriented Thinking and Programming. [S. l.]: Tomáš Bruckner, 2013. Disponível em: https://files.bruckner.cz/be2a5b2104bf393da7092a4200903cc0/PecinovskyOOP.pdf. Acesso em: 31 ago. 2022.

PRISMA DATA, INC. **Quickstart**. [*S. l.*], 2022. Disponível em: https://www.prisma.io/docs/getting-started/quickstart. Acesso em: 2 set. 2022.

REGENTS OF THE UNIVERSITY OF CALIFORNIA. **Human Brain Organogenesis**. [*S. l.*], 2022. Disponível em: https://medschool.ucsd.edu/som/pediatrics/research/labs/muotri-lab/Pages/default.aspx. Acesso em: 27 ago. 2022.

SCHACH, Stephen R. Object-Oriented Software Engineering. 1. ed. Nova Iorque: McGraw-Hill, 2008. ISBN 978-0-07-352333-0. Disponível em: https://www.academia.edu/30989571/Schach_s_object_oriented_software_engineering_7ed_mgh. Acesso em: 2 set. 2022.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Education, 2011. ISBN 978-85-7936-108-1.

THIMBLEBY, Harold. **Technology and the future of healthcare**. Journal of Public Health Research 2013, Swansea, Wales, UK, v. 2, ed. 28, 2013. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4147743/pdf/jphr-2013-3-e28.pdf. Acesso em: 27 ago. 2022.

VISUAL PARADIGM. **UML Class Diagram Tutorial**. [S. l.], 2022. Disponível em: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/. Acesso em: 30 ago. 2022.

VISUAL PARADIGM. **What is Class Diagram?**. [*S. l.*], 2022. Disponível em: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/. Acesso em: 30 ago. 2022.

WEISFELD, Matt. **The Object-Oriented Thought Process**. 3. ed. [*S. l.*]: Pearson Education, 2009. ISBN 978-0-672-33016-2. Disponível em: https://coddyschool.com/upload/Addison_Wesley_The_Object_Orient.pdf. Acesso em: 2 set. 2022.

APÊNDICES

Apêndice A – Requisitos Funcionais

[RF-01] Cadastrar Usuário
[RF-02] Visualizar Usuário
[RF-03] Atualizar Usuário
[RF-04] Remover Usuário
[RF-05] Alterar senha
[RF-06] Autenticar Usuário
[RF-07] Cadastrar Freezer
[RF-08] Visualizar Freezer
[RF-09] Atualizar Freezer
[RF-10] Remover Freezer
[RF-11] Cadastrar Rack
[RF-12] Visualizar Rack
[RF-13] Atualizar Rack
[RF-14] Remover Rack
[RF-15] Cadastrar Caixa
[RF-16] Visualizar Caixa
[RF-17] Atualizar Caixa
[RF-18] Remover Caixa
[RF-19] Cadastrar Frasco
[RF-20] Visualizar Frasco

[RF-21] Atualizar Frasco
[RF-22] Remover Frasco
[RF-23] Cadastrar Célula
[RF-24] Visualizar Célula
[RF-25] Atualizar Célula
[RF-26] Remover Célula

Apêndice B – Casos de uso

Quadro 7 - Caso de uso: Cadastrar Gerente

Caso de uso: Cadastrar Gerente
Descrição de caso de uso: O caso de uso inicia quando o gerente precisa adicionar um novo usuário ao sistema
Atores: Administrador
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Super Usuário já logado no sistema, necessita dos dados do Gerente para cadastro.
Pós-Condições: Cadastro bem-sucedido, novo Gerente criado no sistema.
Fluxo básico: 1. Gerente precisa cadastrar um novo usuário, sistema solicita que informe os dados do usuário; 2. Gerente informa os dados, o sistema valida enviando link para o usuário novo; 3. Usuário autentica entrando no link para o sistema.

Fonte: Autoria Própria (2022)

Quadro 8 - Requisito funcional 02

Caso de uso: trar Freezer
Descrição de caso de uso: O caso de uso inicia quando o gerente precisa adicionar um novo Freezer ao sistema
Atores: Administrador, Gerente
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Ator já logado no sistema
Pós-Condições: Cadastro bem-sucedido, novo freezer criado no sistema.
Fluxo básico: 1. Ator precisa cadastrar um novo freezer; 2. Sistema solicita que informe os dados do freezer; 3. Ator informa os dados; 4. Sistema valida as informações; 5. Freezer cadastrado com sucesso;

Fonte: Autoria Própria (2022)

Quadro 9 - Requisito funcional 03

Caso de uso: Cadastrar Rack
Descrição de caso de uso: O caso de uso inicia quando o ator precisa adicionar um novo Rack ao sistema
Atores: Administrador, Gerente
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Ator já logado no sistema, necessita selecionar um freezer cadastrado
Pós-Condições: Cadastro bem-sucedido, novo rack criado no sistema.
Fluxo básico: 1. Ator precisa cadastrar um novo rack; 2. Sistema solicita que escolha o freezer para inserir o Rack; 3. Ator seleciona o Freezer; 4. Sistema solicita que informe os dados do rack; 5. Ator informa os dados; 6. Sistema valida as informações; 7. Rack cadastrado com sucesso;
Fluxo alternativo: 1. Ator precisa cadastrar um novo rack; 2. Sistema solicita que escolha o freezer para inserir o Rack; 3. Rack extra excede a capacidade do Freezer. Retorna ao passo 2.

Fonte: Autoria Própria (2022)

Quadro 10 - Requisito funcional 04

Caso de uso: Cadastrar Caixa
Descrição de caso de uso: O caso de uso inicia quando o ator precisa adicionar um nova caixa ao sistema
Atores: Administrador, Gerente
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Ator já logado no sistema, necessita selecionar um rack cadastrado.
Pós-Condições: Cadastro bem-sucedido, caixa inserida junto a um rack e criado no sistema.
Fluxo básico: 1. Ator precisa cadastrar uma nova caixa; 2. Sistema solicita que escolha o Rack para inserir a caixa; 3. Ator seleciona o Rack; 4. Sistema solicita que informe os dados da caixa; 5. Ator informa os dados; 6. Sistema valida as informações; 7. Caixa cadastrada com sucesso;
Fluxo alternativo: 1. Ator precisa cadastrar uma nova caixa; 2. Sistema solicita que escolha o Rack para inserir a caixa; 3. Caixa extra excede a capacidade do Rack. Retorna ao passo 2.

Fonte: Autoria Própria (2022)

Quadro 11 - Requisito funcional 05

Caso de uso: Cadastrar Frasco
Descrição de caso de uso: O caso de uso inicia quando o ator precisa adicionar um novo frasco ao sistema
Atores: Administrador, Gerente
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Ator já logado no sistema, necessita selecionar uma caixa cadastrada.
Pós-Condições: Cadastro bem-sucedido, novo frasco criado no sistema.
Fluxo básico: 1. Ator precisa cadastrar um novo frasco; 2. Sistema solicita que escolha o Caixa para inserir a caixa; 3. Ator seleciona a Caixa; 4. Sistema solicita que informe os dados do frasco; 5. Ator informa os dados; 6. Sistema valida as informações; 7. Frasco cadastrado com sucesso;

Fonte: Autoria Própria (2022)

Quadro 12 - Requisito funcional 06

Caso de uso: Cadastrar Célula
Descrição de caso de uso: O caso de uso inicia quando o ator precisa adicionar uma nova célula ao sistema
Atores: Administrador, Gerente
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Ator já logado no sistema, necessita selecionar uma frasco cadastrado
Pós-Condições: Cadastro bem-sucedido, nova célula criada no sistema.
Fluxo básico: 1. Ator precisa cadastrar uma nova Célula; 2. Sistema solicita que escolha o Frasco para inserir a Célula; 3. Ator seleciona o Frasco; 4. Sistema solicita que informe os dados do Frasco; 5. Ator informa os dados; 6. Sistema valida as informações; 7. Célula cadastrada com sucesso;

Fonte: Autoria Própria (2022)

Quadro 13 - Requisito funcional 07

Caso de uso: Atualizar Célula
Descrição de caso de uso: O caso de uso inicia quando o ator precisa atualizar uma célula existente no sistema
Atores: Administrador, Gerente
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Ator já logado no sistema, necessita selecionar uma célula cadastrada
Pós-Condições: Cadastro bem-sucedido, nova célula atualizada no sistema.
Fluxo básico: 1. Ator precisa cadastrar uma nova Célula; 2. Sistema solicita que escolha o Frasco para inserir a Célula; 3. Ator seleciona o Frasco; 4. Sistema solicita que informe os dados do Frasco; 5. Ator informa os dados; 6. Sistema valida as informações; 7. Célula cadastrada com sucesso;

Fonte: Autoria Própria (2022)

Quadro 14 - Requisito funcional 08

Caso de uso: Remover Célula
Descrição de caso de uso: O caso de uso inicia quando o ator precisa remover uma célula no sistema
Atores: Administrador, Gerente
Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Pré-Condições: Ator já logado no sistema, necessita selecionar uma célula cadastrado
Pós-Condições: Cadastro bem-sucedido, célula removida no sistema.
Fluxo básico: 1. Ator precisa cadastrar uma nova Célula; 2. Sistema solicita que escolha o Frasco para inserir a Célula; 3. Ator seleciona o Frasco; 4. Sistema solicita que informe os dados do Frasco; 5. Ator informa os dados; 6. Sistema valida as informações; 7. Célula cadastrada com sucesso;

Fonte: Autoria Própria (2022)