CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ETEC DA ZONA LESTE

Técnico em Desenvolvimento de Sistemas

André Caitano Ferreira

Gustavo Henrique da Silva Pereira

Hugo Reblerson Ferreira dos Santos

Thiago Martins Marques

SCGA:
Sistema de Comunicação e Gerenciamento de Ambientes

São Paulo

2023

André Caitano Ferreira

Gustavo Henrique da Silva Pereira

Hugo Reblerson Ferreira dos Santos

Thiago Martins Marques

SCGA:
Sistema de Comunicação e Gerenciamento de Ambientes

Trabalho de Conclusão de Curso apresentado ao Técnico em Desenvolvimento de Sistemas da Escola Técnica Estadual da Zona Leste, para a disciplina de Planejamento de Trabalho de Conclusão de Curso, administrada pelo Professor Ediney Ciasi Barreto, como requisito parcial para obtenção do título de Técnico em Desenvolvimento de Sistemas.

São Paulo

2023

EPÍGRAFE

“A primeira regra de qualquer tecnologia utilizada nos negócios é que a automação aplicada a uma operação eficiente aumentará a eficiência. A segunda é que a automação aplicada a uma operação ineficiente aumentará a ineficiência.”

BILL GATES

RESUMO

De acordo com Vinha (2016), o clima escolar é compreendido como o conjunto de percepções em relação à instituição de ensino que expõem os fatores relacionados à organização e comunicação, como por exemplo às estruturas pedagógica e administrativa, além das relações humanas que ocorrem no espaço escolar. Dessa forma, problemas na organização e comunicação da programação e agendamento de recursos educacionais, como laboratórios e/ou salas de aula podem acabar resultando em prejuízos no processo de ensino e de aprendizagem, que é salientado por Lagoa (2021), que o clima escolar de um ambiente é um dos fatores importantes de determinação de sucesso ou fracasso da gestão do ensino. Para Pereira (2017), o bom clima existente em uma escola contribui para as relações positivas entre professores, alunos, direção e demais trabalhadores da escola. O objetivo desse trabalho é desenvolver uma plataforma mobile e web que dê suporte à comunicação organizacional, em particular no que tange a programação e agendamento de recursos educacionais, evitando assim problemas de clima organizacional negativo na sala de aula. A Metodologia consiste em utilizar a pesquisa bibliográfica exploratória dos temas entrelaçados nisso, além das linguagens *Hypertext Markup Language* (*HTML*), *Cascading Style Sheets* (*CSS*), *JavaScript* e *Hypertext Preprocessor* (*PHP*), os *Frameworks Laravel* e *Bootstrap* e o Sistema de Banco de Dados *NoSQL Firebase*, todos juntos com a diagramação *Unified Modeling Language* (*UML*), além das tecnologias de *Progressive Web App* (*PWA*) para o desenvolvimento *mobile* do aplicativo.

Palavras-chave: comunicação organizacional; Centro Paula Souza; clima escolar; agendamento de salas; planejamento de aulas; tecnologia.

ABSTRACT

According to Vinha (2016), the school climate is a group of perceptions regarding the educational institution that expose factors related to organization and communication, such as the pedagogical and administrative structures, as well as the human relationships that occur in the school space. Therefore, problems in the organization and communication of the programming and scheduling of educational resources, such as laboratories and/or classrooms, can result in losses in the teaching and learning process, which is emphasized by Lagoa (2021) that the school climate of an environment is one of the crucial factors in determining the success or failure of education management. For Pereira (2017), a good climate in a school contributes to positive relationships between teachers, students, management, and other school workers. The objective of this work is to develop a mobile and web platform that supports organizational communication, particularly regarding programming and scheduling of educational resources, thereby avoiding negative organizational climate problems in the classroom. The methodology consists of using exploratory bibliographic research on the intertwined themes, as well as Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, and Hypertext Preprocessor (PHP) languages, the Laravel and Bootstrap frameworks and the Firebase Database System NoSQL, all together with Unified Model Language (UML), in addition to Progressive Web App (PWA) technologies for the mobile application development.

Keywords: *organizational communication; Centro Paula Souza; school climate; scheduling of rooms; lesson planning; technology.*

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

Application Programming Interface (API)

Cascading Style Sheets (CSS)

Centro Estadual de Educação Técnica e Tecnológica Paula Souza (CEETPS)

Centro Paula Souza (CPS)

Educação Profissional e Tecnológica (EPT)

Escola Técnica Estadual (ETEC)

Extensible Markup Language (XML)

Faculdade de Tecnologia (FATEC)

Fundação de Apoio à Tecnologia (FAT)

Hypertext Markup Language (HTML)

Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol Secure (HTTPS)

JavaScript (JS)

JavaScript Object Notation (JSON)

Model View Controller (MVC)

Not Only SQL (*NoSQL*)

Hypertext Preprocessor (PHP)

Progressive Web App (PWA)

Secretaria de Desenvolvimento Econômico, Ciência, Tecnologia e Inovação (SEDECTI)

Service Worker (SW)

Sistema de Gerenciamento de Banco de Dados (SGBD)

Tecnologia de Informação e Comunicação (TIC)

Unified Modeling Language (UML)

World Wide Web Consortium (W3C)

SUMÁRIO

1 INTRODUÇÃO

No mundo atual, fica claro a importância de uma comunicação boa e compassiva, como por exemplo entre clientes e empresas e com toda a certeza na área educacional, pois como um ensino poderia ser bom sem uma comunicação entre as duas partes principais do pilar da educação: Os Alunos e Professores, além da relação dos professores com os superiores? No entanto, também vale a pena salientar que além desse ponto de vista envolvendo a comunicação, existe também o âmbito organizacional e do clima do ambiente, que segundo Lagoa (2021) é extremamente influenciado por uma cultura organizacional que se desenvolve na mesma.

Com base nisso, é realmente relevante também se ter um aspecto de comunicação flexível, para que dessa forma, a base e o motivo principal do projeto não se quebre e prejudique a outros: a organização e comunicação na procura de laboratórios ou salas disponíveis no ambiente em geral da unidade escolar.

Realizando uma pesquisa exploratória, bibliográfica e das áreas tecnológicas da *Unified Modeling Language* (UML), de Gerenciamento de Banco de Dados e de outras bases, foi planejado um sistema com o foco de desenvolver uma comunicação organizacional mais agradável no âmbito das salas e laboratórios, em que o usuário possa ter um acesso mais amplo ao seu cronograma escolar, junto com a possibilidade de marcar ou reservar um ambiente escolar específico na unidade de uma forma mais maleável e simples, para assim todos que puderem terem informações mais rápidas do campo, evitando assim futuras decepções envolvendo um ambiente não funcional ou já utilizado no momento.

A pesquisa exploratória abrange a área de comunicação organizacional, principalmente envolvida no assunto do ambiente escolar, junto com formas possíveis de manejar isso. Ademais, está incluso o aprofundamento de formas de aprendizagem, junção da tecnologia com a escola e a lógica base de planejamento de aulas. As linguagens *Hypertext Markup Language* (HTML), *Cascading Style Sheets* (CSS), *JavaScript* e *Hypertext Preprocessor* (PHP), o *framework Laravel* e sua estrutura *Model View Controller* (MVC) e *Bootstrap*, junto com a tecnologia de dados de documentos *Firebase,* apresentam-se o suficiente para suprir todas as necessidades do sistema, seja para tornar o site mais acessível para todos, simples,

seguro e prático para o acesso, remoção e visualização de reservas e cronogramas do próprio usuário e do ambiente procurado.

2 REFERENCIAL TEÓRICO

Este capítulo contém todo o embasamento teórico das tecnologias que serão utilizadas para a elaboração do projeto de pesquisa do aplicativo gerenciador de salas e laboratórios.

2.1 Planejamento de aulas nas salas e laboratórios

Geralmente, os professores muitas vezes enfrentam dificuldades ao planejar suas aulas, selecionar conteúdo, métodos e estratégias de avaliação, devido à ampla variedade de opções disponíveis em um contexto extremamente flexível.

De acordo com Leal (2005), o planejamento de aulas é um processo que requer organização, sistematização, previsão, decisão e muitos outros aspectos para garantir a eficácia da ação, independentemente do nível de ensino. Do ponto de vista educacional, o planejamento é um ato político-pedagógico que revela intenções e intencionalidade, expondo o que se deseja realizar e o que se pretende atingir.

Ao se basear no plano de ensino, o professor prepara suas aulas organizando um cronograma que divide o conteúdo programático em módulos para cada aula, contemplando atividades e leituras a serem feitas e discutidas em aula ou em casa pelos alunos. Para cada aula, é necessário ter um plano de aula para facilitar a sistematização das atividades e alcançar os objetivos propostos. Portanto, o planejamento de ensino possui características próprias, pois lida com os alunos, que estão em processo de formação humana (LEAL, 2005).

Assim, pode-se concluir que o ambiente da sala ou laboratório é extremamente importante para o planejamento das aulas. Se houver algum problema com a organização e o cronograma da sala em que o professor irá administrar sua aula, isso pode prejudicar a apresentação do conteúdo, afetando diretamente os alunos e atrasando os planejamentos seguintes, podendo levar assim à perda do conhecimento que seria transmitido aos alunos, o que é prejudicial para o processo de ensino e aprendizagem.

Na figura 1, vemos um exemplo das etapas específicas necessárias para se desenvolver um plano de aula, as quais segundo Daniela (2014) são resumidamente:

1. Tema abordado;
2. Objetivos gerais a serem alcançados;
3. Etapas previstas;
4. A metodologia que o professor usará;
5. Forma de avaliação;
6. Bibliografia.

Figura 1 - Planejamento de uma aula

PLANO DE AULA
CURSO: xxxxxxxxxxxxxxxxx PROFESSOR: xxxxxxx ASSUNTO: xxxxxxx
EMENTA: Xxxxxxxx CARGA HORÁRIA:
OBJETIVO: xxxxxxxx OBJETIVOS ESPECÍFICOS:
1 xxxxxx 2 xxxxxx 3 xxxx
METODOLOGIA: xxxxx
CONTEÚDO PROGRAMÁTICO: 1 xxxxxxxxxx 2 xxxxxxxxxxxx 3 xxxxxxxxxxx
AVALIAÇÃO: xxxxxxxxxxxxx
BIBLIOGRAFIA BÁSICA: Xxxxxxxxxxxxxxx BIBLIOGRAFIA COMPLEMENTAR: Xxxxxxxxxxxxxxxxxxxx

Fonte: Daniela (2014)

2.2 Inovações Tecnológicas nas aulas

Pode-se entender que a presença constante da Tecnologia de Informação e Comunicação (TIC) na vida dos jovens cria um ambiente propício para a incorporação de novas tecnologias na escola. Isso justifica os esforços dos gestores da educação

em investir recursos financeiros em acesso público à rede e em capacitação para professores que não estão familiarizados com a tecnologia, tornando estranho pensar em ambientes de aprendizagem sem conexão com o mundo digital (BIZELLI, 2015).

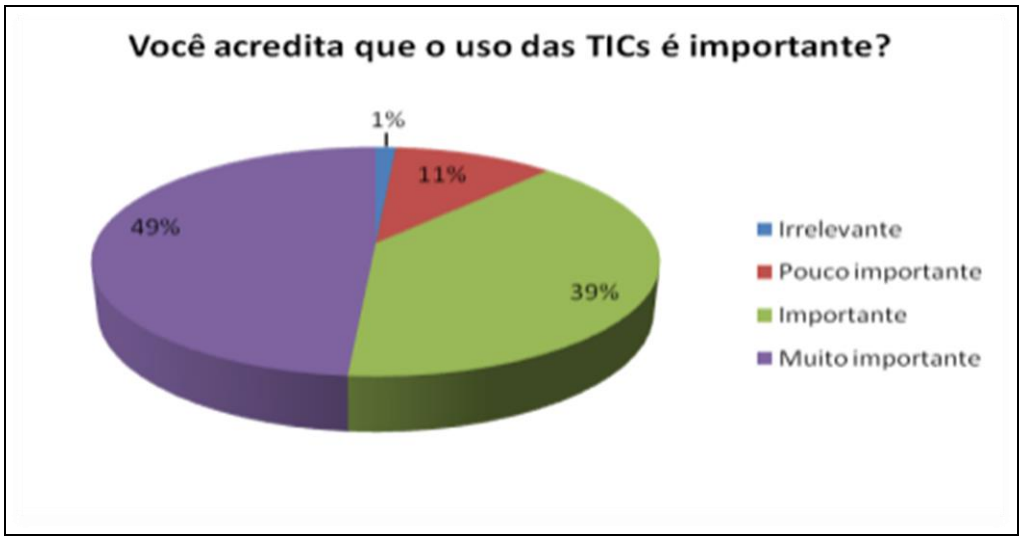
Odorico (2012) salienta que o uso de recursos computacionais pode ser uma ótima forma de melhorar o processo de ensino e aprendizagem, contudo, é necessário que haja contribuição dos programas governamentais, capacitação, apoio técnico e planejamento dos educadores. O professor é peça chave para o desdobramento das TIC em processo de ensino e aprendizagem mais eficaz. Muitos docentes são “imigrantes digitais” com o compromisso de ensinar “nativos digitais”, que aprendem em ritmos e de maneiras diferentes: há choque cultural de gerações na relação professor e estudante, apresentando novas facetas de um mesmo conflito.

Diversos professores se sentem inseguros ao trabalhar com tecnologia da informação com seus alunos, pois sabem que os alunos são mais proficientes do que eles em máquinas e ferramentas digitais. Assim, os professores muitas vezes optam por indicar um tema de pesquisa para seus alunos e dar-lhes liberdade para navegar na rede, o que pode levar a uma mudança de foco. Nesse cenário, é papel da escola, do sistema de ensino e do professor acompanhar e orientar os alunos, fornecendo novos instrumentos para apoiar a aprendizagem (ODORICO, 2012).

Justamente por isso, é fundamental entender como a tecnologia está sendo utilizada pelos educadores em sala de aula para que as políticas públicas possam ser mais efetivas e alinhadas com as práticas de ensino, junto com a utilização dela para manter a unidade da escola mais flexível e acessível. Portanto, melhorar as condições de trabalho do professor é fundamental para melhorar a educação no país (ÂNGELA, 2011).

A figura 2 demonstra essa relevância do assunto, pela pesquisa realizada por José (2015), em que cerca de 88% professores afirmam que o uso de TIC nos espaços escolares é relevante, enquanto 12% o interpretam como de pouca ou nenhuma importância.

Figura 2 - Acredita que o uso de TICs na escola é importante?



Fonte: José (2015)

2.3 Centro Paula Souza

A autarquia denominada Centro Estadual de Educação Técnica e Tecnológica Paula Souza (CEETPS ou CPS), está vinculada à Secretaria de Desenvolvimento Econômico, Ciência, Tecnologia e Inovação (SEDECTI) do Estado de São Paulo e tem como responsabilidade gerenciar as escolas técnicas (Etecs) e faculdades tecnológicas (Fatecs) do estado (SIMON, 2014).

O Centro foi estabelecido em 1969 por meio de um decreto do governador Roberto de Abreu Sodré. Em seu início, o foco do centro estava na formação tecnológica de nível superior, mas posteriormente passou a abranger também a rede de escolas técnicas de nível médio que haviam sido criadas em décadas anteriores no Estado de São Paulo. De acordo com Simon (2014), em 2013 a CEETPS contava com 211 escolas técnicas com 216 mil alunos e 56 faculdades de tecnologia com 65 mil alunos, distribuídos em 161 municípios paulistas, tornando-se a maior instituição estadual do Brasil voltada para a educação profissional articulada ao ensino médio, bem como para a educação tecnológica de nível superior.

Com base no seu local de origem, o Estado de São Paulo tem uma tradição histórica na área de ensino profissionalizante de nível médio, com suas origens remontando ao século 19. A criação da Superintendência da Educação Profissional e Tecnológica (EPT) em 1934 consolidou esse desenvolvimento. A partir de 1980, o Centro Paula

Souza, que originalmente se dedicava ao ensino tecnológico de nível superior, passou a incorporar escolas técnicas de nível médio (SIMON, 2014).

No ano de 2013, o Centro Paula Souza contava com 216 mil alunos matriculados em níveis médio e técnico, sendo que a maioria (166 mil) estava inscrita em cursos técnicos, enquanto 50 mil frequentavam o ensino médio. É importante destacar que havia poucos estudantes matriculados em cursos técnicos integrados com o médio (SIMON, 2014).

Para ingressar nas Etecs, era necessário realizar um exame chamado "vestibulinho", que era administrado pela Fundação de Apoio à Tecnologia (FAT), uma entidade privada vinculada ao CEETPS. No primeiro semestre de 2013, houve 211 mil inscrições para 60 mil vagas disponíveis, resultando em uma taxa de aproximadamente 3,5 candidatos por vaga (SIMON, 2014).

2.4 Análise de problemas no Clima Organizacional e Comunicação

As empresas vivem um constante processo de mudança, e, por consequência de tantas novidades, o comportamento e as atitudes das pessoas acabam-se alterando de forma rápida, sendo necessário ter a ciência de como manter um ambiente saudável e leve no ambiente do trabalho (MAISA, 2009).

Para manter um ambiente de trabalho saudável e produtivo, é importante identificar e solucionar os problemas que afetam o desempenho dos colaboradores. Segundo Maisa (2009), alguns desses exemplos são a falta de motivação, informação e satisfação, que podem ser causados por questões como a comunicação inadequada, a falta de amizade e confiança entre as pessoas. É importante que os gestores fiquem atentos a esses pontos críticos e busquem corrigi-los de forma pacífica, para que possam estabelecer um clima organizacional positivo.

No que se refere à comunicação, é importante que cada indivíduo leve em consideração que seu próprio comportamento, tanto verbal quanto não verbal, influencia as relações com os outros e, consequentemente, a satisfação no trabalho e a motivação para atingir os objetivos da organização.

Contudo, o maior desafio para a investigação em comunicação organizacional está em superar as visões tradicionais e contribuir com uma concepção renovada do ambiente mutável das organizações. Entre as novas vertentes de pesquisa e

investigação, fortalece-se a discussão em torno do caráter intelectual, construtivo e autopoiético da comunicação nas organizações, desde as contribuições originais de Maturana, Varela e *Niklas Luhmann* (CURVELLO, 2002).

Sendo assim, a comunicação é assumida como um fator que facilita dinâmicas de trabalho e, por conseguinte, o desempenho de cada colaborador ou da equipe.

Acrescentando, Marchiori (2018) descreve que é preciso que os profissionais atuem no sentido de “construir fatos” no interior da organização, e não somente apenas pautar as ações na comunicação de fatos que já ocorreram, necessitando que se estimule a possibilidade de situações que permitam a existência de um processo social.

Referente à tecnologia, as mudanças trazidas representam um resgate do receptor como ser ativo no processo comunicativo. O modelo teórico da mensagem que parte de um emissor a um receptor em situação de inferioridade cai por terra. A nova era da interatividade transfere ao antigo receptor o poder de conduzir o processo comunicativo, passando assim a definir o que quer ler, ouvir, ver ou saber (CURVELLO, 2002).

2.5 Pesquisa Exploratória

A pesquisa exploratória, ou estudo exploratório visa obter uma compreensão mais precisa da variável em estudo, sua interpretação e o contexto no qual se encontra inserida. É pressuposto que o comportamento humano é mais bem compreendido dentro de um contexto social. Este tipo de estudo é diferente da maioria dos estudos, pois é realizado na fase de planejamento da pesquisa, como se fosse uma sub pesquisa, com o objetivo de obter informações do Universo de Respostas para refletir a realidade com precisão. (PIOVESAN, 1995).

Dessa forma, de acordo com Piovesan (1995), o objetivo da pesquisa exploratória é evitar que as predisposições do pesquisador afetem as percepções e, consequentemente, a análise dos resultados. Se essa tendência não for corrigida, o pesquisador poderá interpretar a realidade de acordo com sua própria perspectiva, influenciada por suas habilidades técnicas e profissionais. A pesquisa exploratória, portanto, ajuda a controlar esses efeitos distorcidos e permite que a realidade seja percebida objetivamente, sem as interferências do pesquisador.

Por sua vez, Malhotra (2001) descreve que a pesquisa exploratória tem como objetivo compreender as motivações subjacentes de atitudes e comportamentos humanos, como uma técnica comumente usada para gerar hipóteses e identificar variáveis que devem ser incluídas na pesquisa. No entanto, em geral ela ajuda a formar ideias para entender o problema como um todo, enquanto a pesquisa descritiva tem o objetivo de quantificar e analisar os dados coletados estatisticamente, por exemplo.

Dessa forma, os estudos exploratórios são comumente usados para gerar hipóteses e identificar variáveis importantes a serem incluídas na pesquisa. Finalizando, Malhotra (2001) descreve que a coleta de dados qualitativos é a principal metodologia utilizada nos estudos exploratórios, que consiste em um método de coleta de dados não estruturado, baseado em pequenas amostras, assim o objetivo da metodologia é fornecer uma compreensão inicial do problema de pesquisa como um todo.

2.6 HTML

A linguagem *Hypertext Markup Language* (HTML, também chamada de Linguagem de marcação de hipertexto), é usada para descrever e estruturar dados em multimídia de hipertexto. Essa linguagem possui um conjunto completo de padrões que permite incluir gráficos, navegação hipertexto, som, vídeo e outros recursos em páginas da web (PINHEIRO, 1997).

Começando a surgir em 1990, com o protocolo *Hypertext Transfer Protocol* (HTTP), sendo ambos criados por Tim Berners-Lee, foi desenvolvido o HTML 1, a mais simples e com funcionalidades bem limitadas, contudo, ao decorrer desses anos, foram desenvolvidas versões cada vez mais aprimoradas, chegando assim, em 2004 com o HTML 5: a última versão atual de HTML, revolucionando ainda mais o HTML e o que ele pode desenvolver (TORRES, 2018).

Todos os comandos ou sintaxes HTML e suas respectivas propriedades devem estar entre os sinais < e >, que se designam por marcas (*tags*), que quando o browser ou navegador encontra esses sinais, sabe que o que está entre eles não é um texto, mas sim um comando a ser executado, que são fechados com um "/" (*right slash*) para cessar seu efeito (CARVALHO, 1997).

As sintaxes obrigatórias do HTML são seguidas na sequência:

- *doctype*: Uma instrução para o navegador da web que diz em qual versão do HTML a página é escrita, o que garante que a página seja analisada e exibida da mesma maneira por navegadores diferentes.
- *html:* O elemento ancestral de todos os outras *tags html* e da página, chamada de elemento *root* (central). Também é possível e recomendado adicionar a linguagem da página, referente à gramática.
- *head:* Providencia informações gerais (metadados) sobre o documento, incluindo seu título e links para folhas de estilos (CSS) e outras dados da página.
- *meta:* São encarregadas para atribuir determinado tipo ou formato de metadados no documento/página HTML.
- *title*: O título da página que será mostrado para o usuário na guia do navegador.
- *body:* A parte principal do documento, é nela que ficam os elementos que dão a cara e visualização de elemento ou *tags* da página, como o próprio nome diz, o corpo da página.

A figura seguinte demonstra o formato da estrutura atual no HTML5:

Figura 3 - Estrutura básica HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Documento</title>
</head>
<body>

</body>
</html>
```

Fonte: Autoria Própria, 2023.

Dentro do body, podem ser utilizadas determinadas *tags*, que são o caso por exemplo, de:

- *h1-h6 (headers)*: Cabeçalhos da página, que normalmente são utilizados para o título ou subtítulos de uma página. Eles têm um formato padrão de h1 (maior) até o h6 (menor).

- *p*: Os parágrafos de uma página, que são utilizados para o formato de texto puro para descrição e outros derivados, que pode ser interpretado como origem de muitas outras *tags*, como por exemplo o *span*, *i*, *b*, *strong*, *abbr*, *code, span, dd* e *dl* (referente à estilos) e o *a* (elemento utilizado para adicionar links na página).
- *div*: Define uma seção no documento, e é pai e origem de diversos outros elementos *HTML* que são utilizados no código, como alguns citados anteriormente e até o próprio elemento com tags diferentes para a semântica.
- form: Se desenvolve um formulário na página, em que dentro fica diversos itens (ou *tags*) necessários para o desenvolvimento do mesmo e das escolhas do usuário, como o *input* (campo de inserção personalizável), *label* (nome do campo), que fazem o papel essencial de escolha dos campos de inserções, legenda e o *submit* (envio) pelo *button* (botão) no final do *form*.

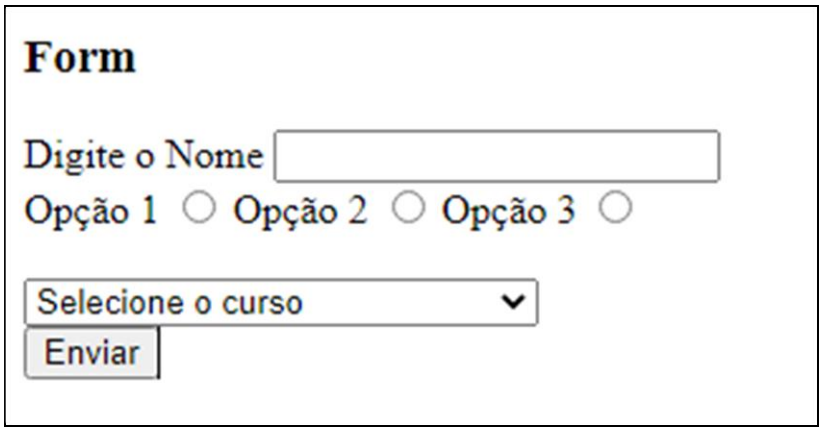
Na figura seguinte temos um exemplo de um código do *body*, enquanto na figura 5 temos o resultado em imagem da junção da figura 3 com a 4:

Figura 4 - Estrutura de um Body

```
<div class="container">
    <div class="container form">
        <h3>Form</h3>
        <form id="formulario">
            <label for="texto">Digite o Nome
                <input type="text" name="texto" id="texto">
            </label>
            <br>
            <div>
                <label for="opcao1">Opção 1
                    <input type="radio" name="opcao" value="opcao1" id="escolha1">
                </label>
                <label for="opcao2">Opção 2
                    <input type="radio" name="opcao" value="opcao2" id="escolha2">
                </label>
                <label for="opcao3">Opção 3
                    <input type="radio" name="opcao" value="opcao3" id="escolha3">
                </label>
            </div>
            <br>
            <div>
                <select name="curso">
                    <option value="curso">Selecione o curso</option>
                    <option value="DS">Desenvolvimento de Sistemas</option>
                </select>
            <br>
            <input type="submit"></button>
        </form>
    </div>
    <p id="mensagem"></p>
</div>
```

Fonte: Autoria Própria, 2023.

Figura 5 - Resultado do Código puro HTML



Fonte: Autoria Própria, 2023.

Com base nisso tudo, podemos perceber que o HTML5 transformou seus códigos, deixando-os bem mais legíveis em comparação às suas versões anteriores, contudo, como se pode ver, HTML5 e suas outras versões não conseguem estilizar as

páginas web de uma forma agradável e intuitiva para seus usuários, e para esse tipo de trabalho é então necessário utilizar o CSS (FÁBIO, 2011).

2.7 CSS

Em 1995 o *Cascading Style Sheet* (CSS) foi desenvolvido pela *World Wide Web Consortium* (W3C), um grupo de empresas do ramo da informática. A linguagem de estilos ganhou muito destaque entre 1997 e 1999, neste período ficou conhecido por grande parte dos programadores. De início, o HTML era a única linguagem utilizada para criar sites (VANDRÉ, 2019).

Quando o HTML começou a ganhar popularidade, algumas *tags* (comandos) foram criadas pelos navegadores para facilitar o uso dessa linguagem, assim a W3C cria o CSS, colocando à disposição dos desenvolvedores Web, utilizando o principal conceito como separação de interesses (*separation of concerns*), fazendo com que os próprios navegadores recebessem essas *tags* de construção do conteúdo, e cuidassem de juntar e alinhar todos, assim tendo uma facilidade ampla na linguagem (LEWIS, 2010).

Sendo utilizado pelos programadores, o CSS controla por exemplo: as opções de margem (*margin*), linhas, cores (*color*), alturas (*height*), larguras (*width*), imagens, posicionamento, texto (*font* e *text*), bordas (*border*), espaço (*padding*) e derivados, sem necessidade de programar em HTML. Além disso, o CSS tem também diversos códigos prontos, feito pela própria comunidade, permitindo aos usuários pouparem tempo criando códigos muito comuns, e facilitando a vida de quem trabalha na área de informática (ÁLVAREZ, 2017).

Na figura 6, é mostrado um código utilizado no HTML para importar um arquivo CSS (que é colocado no *header*). O elemento *link* define o relacionamento entre o documento HTML e um arquivo ou imagem externa, sendo muito utilizado para conectar uma folha de estilo à página.

Figura 6 - Adicionando arquivos CSS no HTML

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Documento</title>
</head>
```

Fonte: Autoria Própria, 2023.

Uma sintaxe CSS é por uma regra com 3 itens fundamentais para definir um estilo, que são:

- Seletor: Vincula um elemento do documento HTML a declaração CSS, ela é formada pela propriedade e o valor e pode envolver alguns pequenos cálculos e técnicas distintas para selecionar determinado elemento para estilizá-lo com base dos atributos internos da seleção.
- Propriedade: define uma característica visual para o elemento HTML “selecionado” pelo seletor. Pode-se dizer que a propriedade é o que será estilizado enquanto seu valor é a forma.
- Valor: atribui valor a propriedade escolhida para o elemento selecionado. O valor levará em conta a propriedade selecionada antes, ou seja, o que pode determinar o formato do valor será essa mesma propriedade definida anteriormente.

A seguinte figura demonstra o formato utilizado nessa regra e formato fundamental do CSS.

Figura 7 - Formatação fundamental de um código CSS

```
/* h1: seletor, {}:corpo da regra */
h1{
    /*propriedade: valor;*/
    color: #333;
    font-size: 12px;
    text-align: center;
}
```

Fonte: Autoria Própria, 2023.

Algumas formas de adicionar seletores são:

- Pelo Nome da *tag.*
- Pelo *id*, declarado como um atributo de uma *tag* específica (que em geral definido ser um valor único).
- Pela classe, declarado também como o *id*, e normalmente sendo utilizado em diversos elementos de uma *tag* para o que se deseja em estilização.

A figura 8 e 9 apresenta um código CSS responsável por estilizar o HTML exibido na figura 5, o tornando mais agradável e apresentável ao usuário, sendo possível ver o resultado dessa estilização na figura 10.

Figura 8 - Código CSS de um site (1)

```css
body{
    background: rgb(153, 192, 230);
}

.container{
    display: grid;
    justify-items: center;
    align-items: center;
}

#cabecalho-inicial{
    color: aquamarine;
    font-weight: bolder;
}
h3{
    font: 2.5rem bolder;
    font-family: Arial, Helvetica, sans-serif;
}

.form{
    background-color: bisque;
    padding: 1rem;
}
```

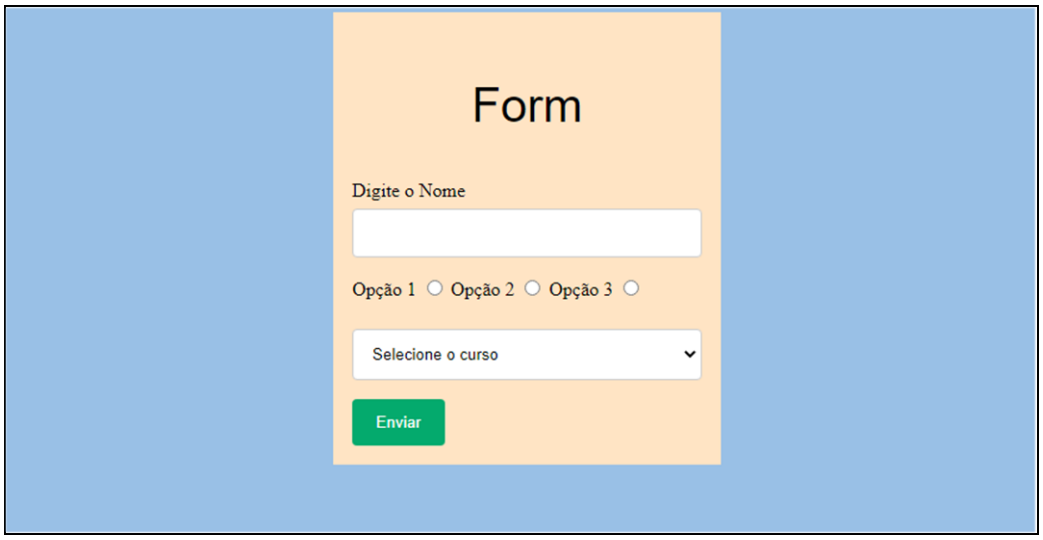
Fonte: Autoria Própria, 2023.

Figura 9 - Código CSS de um site (2)

```css
input[type=text], select, textarea {
    width: 100%;
    padding: 12px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
    margin-top: 6px;
    margin-bottom: 16px;
    resize: vertical;
}
input[type=submit] {
    background-color: #04AA6D;
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

Fonte: Autoria Própria, 2023

Figura 10 - Resultado dos Códigos HTML e CSS



Fonte: Autoria Própria, 2023.

2.8 JavaScript

Também chamado de JS, *JavaScript* é uma linguagem *client-side*, ou seja, ela é executada a partir do cliente, nesse caso, o programador. Não é necessário, diante disso, acessar servidores, o que geralmente envolve procedimentos de complexidade maior. Sendo assim, o profissional pode trabalhar a partir dos navegadores tradicionais existentes no mercado. (IVAN, 2019).

O *JavaScript* permite criar pequenos programas que podem ser incorporados diretamente no código de uma página HTML. Eles podem gerar números, processar dados, verificar formulários, alterar valores de elementos HTML e criar elementos HTML. Quando todas essas operações são realizadas no computador do cliente, a comunicação com o servidor é reduzida e o tempo de processamento depende apenas da velocidade do computador do usuário, não mais da rede (IVAN, 2019).

Grandes empresas, como o Google, utilizam *o JavaScript* em quase todos os seus aplicativos, incluindo o site de busca *Google*, o *Gmail* e o *Google Maps*. Isso se deve à capacidade do *JavaScript* de tornar as aplicações mais interativas e dinâmicas (PRESCOTT, 2016).

Existem muitas tags e elementos no *Javascript* para auxiliar o desenvolvedor na criação de um site para web mais dinâmico e interativo, algumas dessas tags e comandos são específicos para operar em navegadores web, enquanto outras fazem parte do próprio *Javascript*. Abaixo estão algumas das tags e elementos essenciais do *JavaScript*:

<*script*>*:* Esta tag é usada para incluir código *JavaScript* em um documento HTML. Dentro dessa tag, você pode escrever código *JavaScript* diretamente ou apontar para um arquivo Javascript externo usando o atributo "src".

alert: É um comando em *JavaScript* é uma função que exibe uma mensagem na forma de um diálogo modal na tela do usuário. O *alert* é usado principalmente para fornecer informações ou notificações importantes ao usuário de uma página da web.

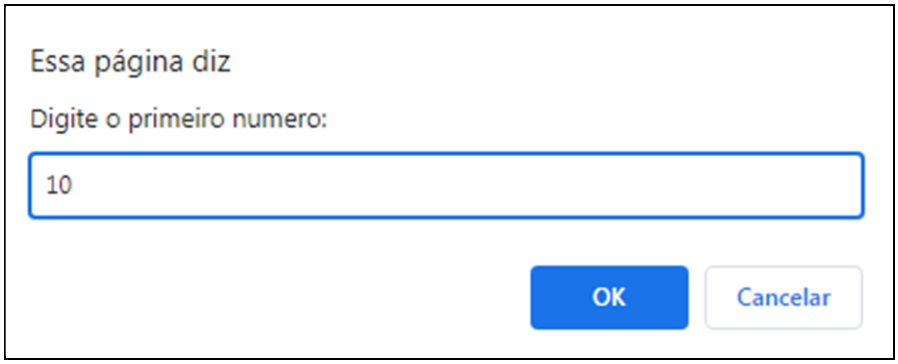
Na figura 11, é demonstrado como adicionar o codigo JS no HTML (no *Head* utilizando a *tag Script*) que retorna a soma de dois numeros inseridos pelo usuário na caixa de inserção de dados. Na figura 12 é demonstrado a primeira tela que é exibida ao usuário perguntando o primeiro numero, na figura 13 segue o mesmo formato da anterior, só que perguntando o segundo numero ao usuário e por fim na figura 14 é apresentado o resultado final da aplicação, exibindo o resultado da soma dos dois numeros.

Figura 11 - Código da soma de 2 números

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        const n1 = parseFloat(prompt("digite o primeiro numero: "));
        const n2 = parseFloat(prompt("digite o segundo numero: "));

        alert("A soma dos numero é: " + (n1 + n2));
    </script>
</head>
<body>

</body>
</html>
```

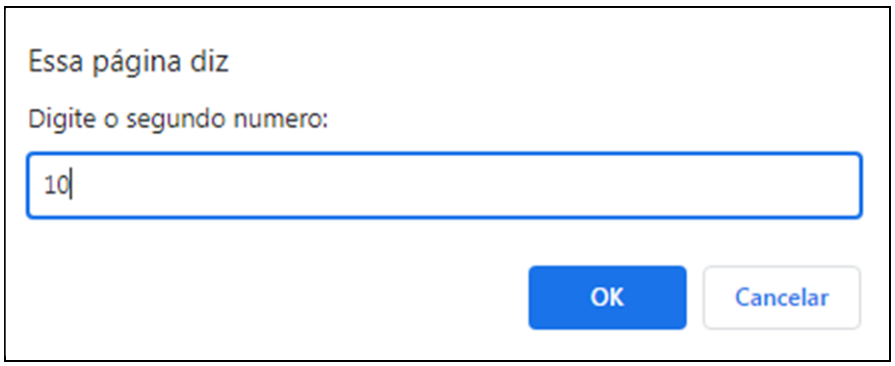
Fonte: Autoria Própria, 2023.

Figura 12 - Exemplo da aplicação pedindo primeiro número através do prompt



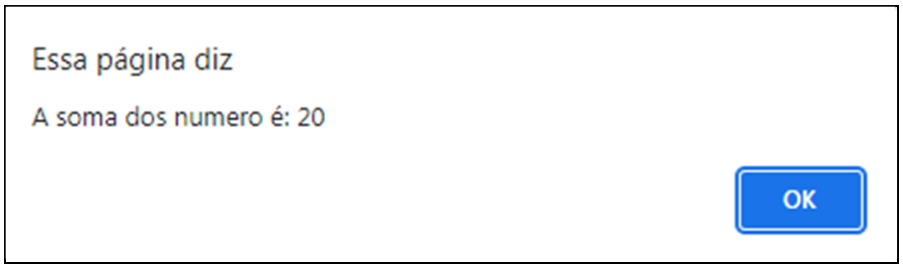
Fonte: Autoria Própria, 2023.

Figura 13 - Exemplo da aplicação pedindo segundo número através do prompt



Fonte: Autoria Própria, 2023.

Figura 14 - Exemplo do resultado da aplicação somando os números



Fonte: Autoria Própria, 2023.

O *JavaScript* possui três formas principais de declarar variáveis: var, let e const, sendo que a primeira na versão atual de JS (*ECMAScript* 6) acaba sendo a menos utilizada de todas. Enquanto a var pode ser declarada novamente e atualizada, let e const são declaradas somente uma vez, sendo que uma pode ser redeclarada com um novo valor (*let*) e outra que simplesmente não pode ser redeclarada (*const*).

Além disso, o *JavaScript* permite acessar e modificar elementos HTML, conhecido como *Document Object Model* (DOM), por meio de funções e atributos específicos. Por exemplo, é possível adicionar funcionalidades a um botão ao clicar nele.

O código da figura 15 é um exemplo de como usar o DOM em conjunto com a lógica de programação para validar formulários. O código *JavaScript* utiliza a lógica da figura 5, junto com o documento HTML e o estilo desenvolvido em CSS anteriormente.

Figura 15 - Código JS para o formulário

```
const form = document.querySelector('#formulario');
form.addEventListener('submit', verificarForm)

function verificarForm(e){
    e.preventDefault();
    const radioOpcao = document.querySelector('input[name="opcao"]:checked');
    const selectOpcao = document.querySelector('select[name="curso"]').value;
    const inputTexto = document.querySelector('#texto');
    const msg = document.querySelector('#mensagem');
    msg.innerHTML = '';

    if(!radioOpcao){
        msg.innerHTML += 'valor nulo no radio <br>';
    }

    if(selectOpcao == 'curso'){
        msg.innerHTML += 'valor nulo no curso <br>';
    }

    if(inputTexto.value.trim() === ''){
        msg.innerHTML += 'valor nulo no input text <br>';
    }

    if(msg.innerHTML == ''){
        msg.innerHTML = 'Olá, seja bem vindo ' + inputTexto.value +'! <br> você escolheu a '
        + radioOpcao.value + ' <br> você é do curso ' + selectOpcao;
    }
   form.reset();
}
```

Fonte: Autoria Própria, 2023.

Na figura 16 é possível visualizar o resultado se os campos não forem preenchidos e o usuário tentar enviar o formulário do lado esquerdo. Se tudo ocorrer como esperado veremos a impressão do lado direito, ou seja, as informações que foram digitadas no formulário pelo usuário.

Figura 16 - Possíveis resultados

Fonte: Autoria Própria, 2023.

2.8.1 JSON

O *Javascript Object Notation* (*JSON*) é uma formatação leve de troca de dados, que é fácil de um humano e uma máquina visualizarem e interpretarem. De acordo com Ben (2020), o *JSON* em poucas palavras é uma representação textual definida por um pequeno conjunto de regras, em que na especificação é definido que os dados podem ser estruturados em uma coleção de pares nome/valor ou em uma lista ordenada.

De acordo com Machado (2017), seguindo a sequência de *tokens* do padrão *Unicode* (texto), o *JSON* permite que os dados sejam armazenados em quatro tipos primitivos diferentes (*strings*, números, inteiros e booleanos), além de suportar a criação de sub objetos utilizando as estruturas de objetos e *arrays*, possibilitando a representação de dados de forma diversificada.

Abaixo vermos um exemplo de um *JSON*, em que tudo é armazenado dentro de uma chave (escopo de objeto), em que existem diversos atributos dos mais variados tipos e de *arrays* (escopo []), como demonstrado na figura 17.

Figura 17 - Exemplo do formato JSON

```
{
    "linguagens":[
        "Python",
        "PHP",
        "Java"
    ],
    "id": "D313EDS42OP213JDFI2",
    "valor": 77,
    "tipo": "json",
    "objeto":{
        "Obj1": "valor1",
        "Obj2": "valor2"
    },
    "status": true
}
```

Fonte: Autoria Própria, 2023.

Assim é possível entender o motivo da utilização dele, principalmente em *Application Programming Interfaces* (API), pois como Caetano (2018) descreve, o objeto JS utiliza uma estrutura de arquivos composta de propriedade-valor que permite uma visível redução de bytes ao arquivo, possibilitando o intercâmbio de dados com uma menor exigência de banda, sendo extremamente acessível, rápido e utilizado muito, não importando a linguagem utilizada para coletá-la.

2.9 PWA

Segundo Crispiniano (2021), o *Progressive Web App* (PWA), é uma estrutura de desenvolvimento que tem como objetivo resgatar para aplicações web, acessadas por navegadores moveis, a mesma capacidade (leve e responsiva) observada nos aplicativos nativos.

O PWA surgiu em 2015, desenvolvido por uma equipe do *Google Developers* composta por nomes importantes como a designer *Frances Berriman* e o engenheiro do *Google Chrome Alex Russell* (TRINDADE, 2018).

De acordo com Trindade (2018), ao utilizar a estrutura de PWA, é possível obter diversas funcionalidades extras em dispositivos móveis, como a possibilidade de operar *offline*, notificações *push*, gerenciamento de memória, carregamento mais rápido, design e performance semelhantes aos de aplicativos nativos. Além disso, a interface amigável e a possibilidade de acesso em tela cheia sem a necessidade da barra de endereços e pesquisa podem fazer com que os usuários não distingam um PWA de um aplicativo nativo instalado via *app store*.

Resumidamente, *os* PWAs oferecem uma solução econômica e eficiente para empresas que desejam migrar sua presença *web* para dispositivos móveis sem o alto custo de desenvolvimento e manutenção de aplicativos nativos para cada plataforma. A utilização do conceito PWA permite que a aplicação seja implementada em uma única linguagem de programação (*JavaScript)* e possa ser usada em várias plataformas, incluindo *iOS* e *Android*, tornando o desenvolvimento mais simples e rápido, reduzindo significativamente o custo e o tempo de produção (TRINDADE, 2018).

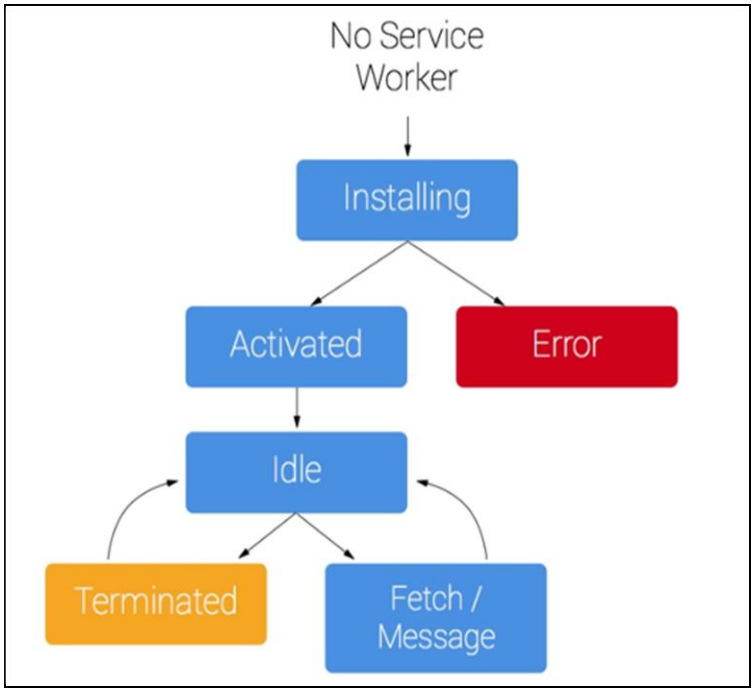
2.9.1 Service Worker

De acordo com Mazzarolo (2021), um *service worker* (SW) é um tipo especial de *worker* baseado em eventos, no qual é registrado um determinado path e origem. Na prática, ele é um arquivo *JavaScript* que contém o controle das páginas do site ao qual ele está associado, interceptando e modificando requisições e a navegação em si. Ele também faz caches dos recursos trafegados de forma abrangente e granular, visando oferecer controle total sobre e como a sua aplicação se comporta em determinadas situações (o exemplo mais óbvio, naturalmente, é quando não há conexão de rede disponível).

Por questões de segurança, *SWs* funcionam apenas em sites servidos via *Hypertext Transfer Protocol Secure* (HTTPS), devido a possibilidade de modificar requisições em um domínio desprotegido contra-ataques (MAZZAROLO, 2021).

Na figura seguinte vemos o ciclo de vida de um *service worker,* em que é chamado pelo código e instalado. Se tudo ocorrer como se deve, ele será ativado, mandará uma mensagem confirmando isso e será fechado (GAUNT, 2021).

Figura 18 - Ciclo de Vida de um Service Worker



Fonte: Gaunt (2021)

2.9.2 Manifest

De acordo com Trindade (2018), certos arquivos, conhecidos como manifesto de aplicativos *Web*, são utilizados para informar ao navegador como exibir o conteúdo de uma página como um aplicativo no sistema operacional. Esses arquivos contêm informações essenciais, tais como o nome, ícone e cor do tema do aplicativo, bem como preferências avançadas, incluindo orientação desejada e atalhos de aplicativos, e podem incluir metadados de catálogo, como capturas de tela.

Qual é a importância de um manifesto dentro de um aplicativo PWA? É de real importância que cada PWA tenha um único manifesto por aplicativo, que geralmente é armazenado na pasta raiz e associado em todas as páginas HTML em que o PWA pode ser instalado. A extensão oficial desse manifesto é *webmanifest,* permitindo que

ele seja nomeado como algo do tipo "webmanifestapp.webmanifest" (TRINDADE, 2018).

Na figura 19, vemos alguns atributos adicionados no arquivo e o que cada um faz, como o tema de cor da barra superior da tela do celular (*theme_color*), a cor de fundo (*background_color)*, o ícone (*icon*), e claro, o nome da aplicação PWA (*name*), enquanto na figura 20 temos uma amostra de um *manifest json*.

Figura 19 - Tela inicial de um PWA e sua conexão com o Manifest

Fonte: Autoria Própria, 2023.

Figura 20 - Exemplo de um Manifesto

```
{
    "theme_color": "#eee",
    "background_color": "#86A7f5",
    "display": "fullscreen",
    "start_url": "/index.php",
    "name": "SCGA - Comunicação e Gerenciamento",
    "short_name": "SCGA",
    "description": "Sistema de Comunicação e Gerenciamento de Ambientes (Salas e Labs)",
    "icons": [
        {
            "src": "img/icon-192x192.png",
            "sizes": "192x192",
            "type": "image/png"
        },
        {
            "src": "img/icon-256x256.png",
            "sizes": "256x256",
            "type": "image/png"
        },
        {
            "src": "img/icon-384x384.png",
            "sizes": "384x384",
            "type": "image/png"
        },
        {
            "src": "img/icon-512x512.png",
            "sizes": "512x512",
            "type": "image/png"
        }
    ]
}
```

Fonte: Autoria Própria, 2023.

2.10 PHP

Seguindo as informações apresentadas por Pablo (2015), o *Hypertext Preprocessor* (PHP) é uma linguagem de *script* de código aberto amplamente utilizada no desenvolvimento *web*, com possibilidade de integração ao HTML. Sua popularidade é atribuída à facilidade de uso na criação de aplicações dinâmicas, compatibilidade com a maioria dos bancos de dados existentes e seu conjunto de funções que possibilitam a criação de portais simples e aplicações de negócios complexas por meio de uma estrutura flexível de programação. Além disso, a utilização da Orientação a Objetos, juntamente com boas práticas de programação, permite um desenvolvimento sustentável em termos de ritmo das aplicações.

O PHP é uma linguagem altamente versátil, permitindo a disponibilização de aplicações na internet de maneira facilitada, com uma curva de aprendizado baixa.

Com a crescente demanda por sistemas baseados na internet, muitas empresas têm sido motivadas a disponibilizar suas aplicações online. Dentre as diversas opções disponíveis, o PHP é uma das linguagens mais populares e amplamente conhecidas, graças à sua facilidade de desenvolvimento e baixo nível de complexidade na aprendizagem (FERNANDO, 2009).

Uma das vantagens dele é que diferente do Javascript, ele pode diretamente processar uma página HTML, e no mesmo arquivo, utilizar sua lógica da linguagem. Isso ocorre, pois, uma das propriedades dele faz com que se tenha essa opção de integração, que é sua declaração: o *<?php* (início) e *?>* (fim), que serve para abrir e fechar o código interno do PHP, podendo ser feito facilmente uma página com HTML, que por dentro contém o processamento do servidor (PHP).

Além disso, como mostrado na figura 21, é possível utilizar o método *Post*, para redirecionar ao local especificado do *action* os valores inseridos no input do *form*.

Figura 21 - Método POST e action

```
<form id="formulario" method="POST" action="./php/verificarForm.php">
```

Fonte: Autoria Própria, 2023.

Abaixo vemos uma validação de formulário utilizando a linguagem e o código da figura 22, que utiliza:

- *$_POST:* Resumidamente retorna o valor do atributo coletado no formulário pelo método e atributo *name* do elemento, utilizando a sintaxe descrita na figura seguinte.
- *$_SESSION*: Como o próprio nome diz, ela funciona como uma variável de sessão, isto é, a guia atual de determinado usuário, assim que é fechada ou o tempo limite é finalizado, volta a se tornar nula (caso tenha um valor), sendo uma variável temporária e global. Para que 2 arquivos possam visualizá-las no seu formato que estiver, ambos devem iniciar a sessão no *php*, utilizando *session_start()* no início do arquivo.

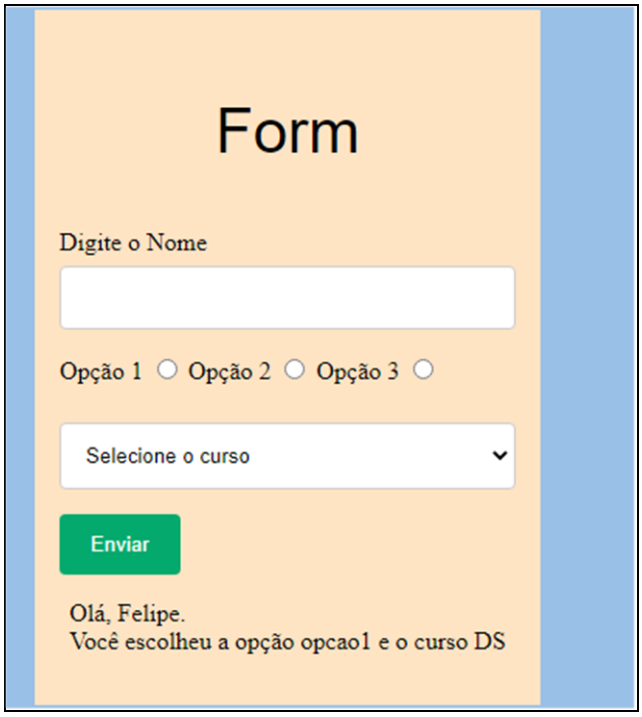
Finalmente, se tudo der certo, será impresso um texto de sucesso na tela, como a figura 23, se não, será impresso “atributo x está nulo ou não selecionado” como na figura 24. O código principal do index utiliza a lógica da figura 5, junto com o estilo desenvolvido em CSS anteriormente.

Figura 22 - Script PHP para validação do formulário

```php
<?php
session_start();

//$_POST['atributo'];
//$_SESSION['atributo'];
$texto = $_POST['texto'];
$opcao = $_POST['opcao'];
$curso = $_POST['curso'];
if(strlen($texto) != 0 && $opcao != '' && $curso != 'curso'){
    $_SESSION['mensagem'] = "Olá, " . $texto . ". <br> Você escolheu a opção " .
    $opcao . " e o curso " . $curso;
} else if(strlen($texto) == 0){
    $_SESSION['mensagem'] = "atributo texto está nulo";
} else if($opcao == ''){
    $_SESSION['mensagem'] = "atributo opcão não está selecionado";
} else if($curso == 'curso'){
    $_SESSION['mensagem'] = "atributo curso não está selecionado";
}

//retorna para a guia inicial
header('location: ../testePhp.php');

//para imprimir na tela a mensagem, no "testePhp.php" é utilizado uma verificação para
//confirmar se o valor não está nulo em um p
/*<p> <?php if(isset($_SESSION['mensagem'])): echo $_SESSION['mensagem']; endif ?></p>*/
```

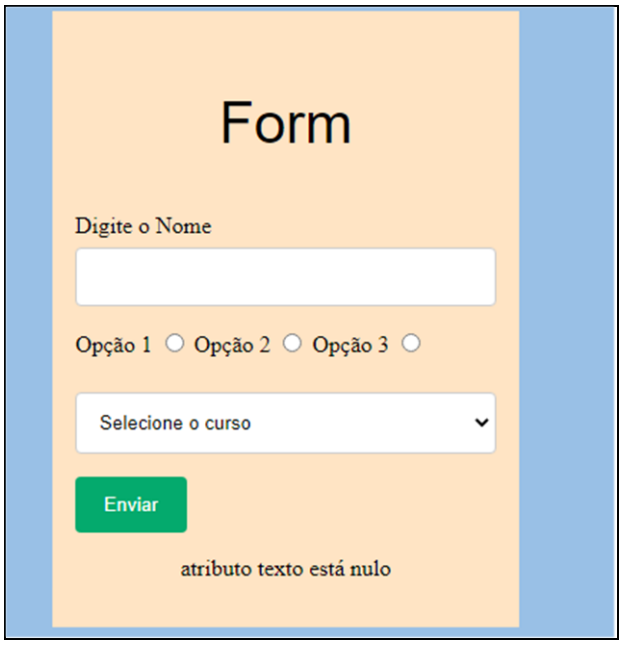
Fonte: Autoria Própria, 2023.

Figura 23 - Retorno de sucesso na validação do Script PHP



Fonte: Autoria Própria, 2023.

Figura 24 - Retorno de falha na validação do Script PHP



Fonte: Autoria Própria, 2023.

2.10.1 Composer

De acordo com a própria documentação do site, o *Composer* é uma ferramenta para gerenciamento de dependência em PHP. Ele permite que você declare as bibliotecas de que seu projeto depende e que irão gerenciar a instalação e atualização desses pacotes para o programador.

Diferentemente de gerenciados de pacotes do Linux, com YUM e APT, ele serve para gerenciar dependências e pacotes em nível de projeto e não globalmente, ou seja, ao adicionar uma biblioteca de autenticação para um projeto, essa biblioteca será incluída somente nesse, sem afetar os demais projetos do servidor ou computador local (GABARDO, 2017).

Na figura abaixo vemos um formato padrão de um *composer json* (arquivo incluído ao executar o comando *composer init*).

Figura 25 - Formato Composer.json

```
{
    "name": "gustavo/exemplo-composer",
    "description": "Pequeno projeto mostrando a funcionalidade do composer",
    "type": "project",
    "license": "MIT",
    "autoload": {
        "psr-4": {
            "Gustavo\\ExemploComposer\\": "src/"
        }
    },
    "authors": [
        {
            "name": "Gustavo-Silva",
            "email": "gustavohenrique123@gmail.com"
        }
    ],
    "require": {}
}
```

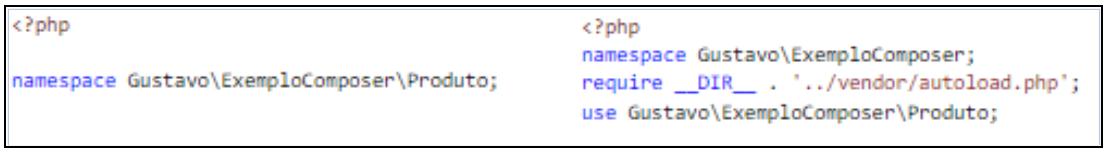
Fonte: Autoria Própria, 2023.

Vemos, dessa forma, pequenas personalizações, como nome e descrição do projeto, licença, o autor, o atributo *require* (onde fica o nome e versão das dependências instaladas no projeto) e o campo *autoload*, junto com *psr-4* (uma norma geral do PHP)

O *autoload*, como o nome sugere (carregamento automático) coleta o nome declarado dentro da *psr-4* para servir como base de importação e exportação mais dinâmica do projeto, precisando somente utilizar o valor da pasta *root* (*src/)* com o nome descrito do lado junto com o caminho do nome da pasta de um arquivo PHP para fazer isso sem muitos problemas. Isso é muito importante para o controle e conexão do seu projeto com outros de dependência, que ficam na pasta *vendor* do projeto.

Na figura seguinte vemos um exemplo em prática, em que existe um arquivo dentro de uma pasta Produto e do *src*, enquanto existe um somente na pasta *root* do *composer*, que deseja importar esse arquivo. É utilizado então o *namespace* para se declarar o local (uma exportação), e depois no index dar o *require n*o *autoload* para assim importar (*use*) o arquivo que está no determinado caminho (lembrando que o nome do src no projeto é o descrito na Figura anterior, o *Gustavo/ExemploComposer*).

Figura 26 - Exemplo da Função do Autoload



```php
<?php

namespace Gustavo\ExemploComposer\Produto;
```

```php
<?php
namespace Gustavo\ExemploComposer;
require __DIR__ . '../vendor/autoload.php';
use Gustavo\ExemploComposer\Produto;
```

Fonte: Autoria Própria, 2023.

Gerenciar pacotes e dependências como visto é rotina no desenvolvimento web, e existem diversos gerenciadores em outras linguagens para com o objetivo do citado.

2.11 Modelo MVC

Incorporado em projetos pela primeira vez no *Smaltalk*, em 1970, o *Model-View-Controller* (MVC), sendo em tradução para português “Modelo, Visualização e Controlador”. Ele é um padrão de design de projetos que separa as camadas de lógica e apresentação oferecendo diversas vantagens no ciclo de vida do projeto, do aspecto de organização do código com a separação do HTML da lógica e das regras de negócio (GABARDO, 2017).

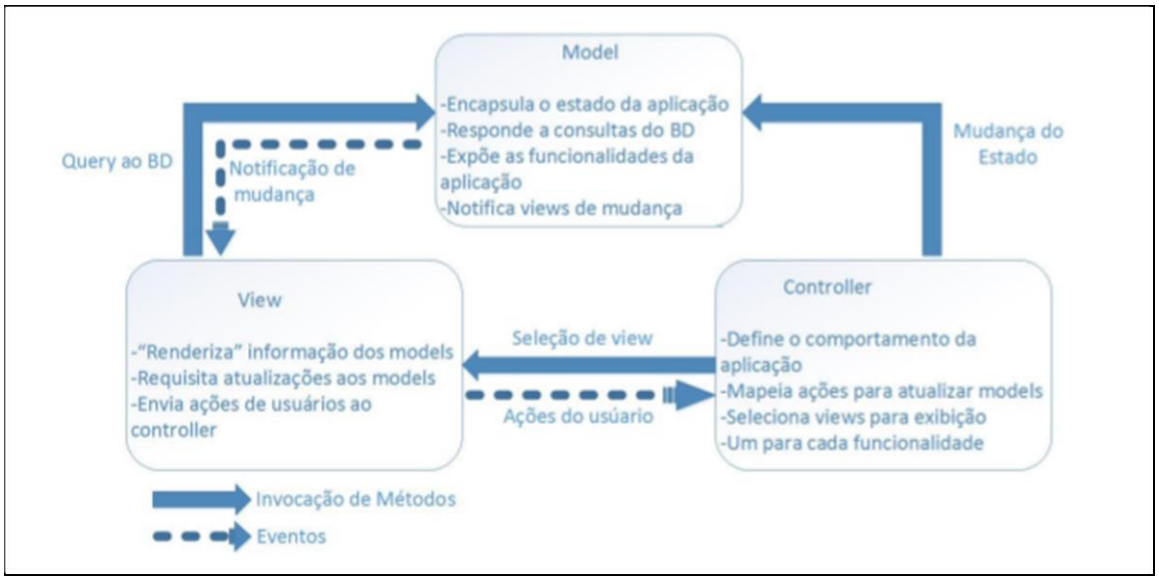
Cada um dos três tem as suas utilidades, que segundo Lemos (2013) são:

- *Model:* contém a comunicação com os dados armazenados que serão visualizados na *View*, podendo estar armazenado em um banco de dados, arquivo XML/JSON ou em qualquer meio para tal funcionalidade. É somente nele que as operações de *Create, Read, Update* e *Delete* (CRUD), podem ocorrer.
- *View*: É a camada de apresentação da aplicação, o que será visualizado pelo usuário final, não importando quais dados e de qual lugar tenham vindo, mas sim de como elas serão exibidas essas informações.
- *Controller:* Responsável por administrar todo o fluxo da aplicação, sendo o que move uma aplicação, que utiliza a lógica implementada para trabalhar com os dados *View* e resolve uma operação especifica no caso do *Model*.

A maioria dos *frameworks* utiliza o paradigma ou padrão de design MVC, como por exemplo o *Spring* (*Java*), *Django* (*Python*), *Angular* e *Next* (*JavaScript*), *Symfony* (PHP), e claro, o *framework* principal que será utilizado no projeto, o *Laravel*.

Na figura abaixo, vemos um exemplo mais aprofundado do modelo MVC, seguindo o que Lemos (2013) exemplificou anteriormente.

Figura 27 - Exemplo do padrão MVC



Fonte: Lemos (2013)

2.12 Frameworks

Os frameworks servem a diversos propósitos, como reduzir ou eliminar a reescrita de código-fonte por meio de reutilização de métodos, classes e funções, forçar a adoção de um padrão ao adotar um determinado tipo de design de projetos. O *Model-View-Controller* (MVC), por exemplo, oferece diversos recursos de uso comuns com validações, criação de formulários, autenticação e outros recursos já prontos ou semiprontos para que não seja necessário reescrever tais funções a cada novo projeto (GABARDO, 2017).

Os *frameworks* utilizados nesse projeto são o *Bootstrap* (CSS/JS) e *Laravel* (PHP).

2.12.1 Bootstrap

Bootstrap é um *framework front-end*, que disponibiliza código fonte para a criação de interfaces *Web*. É considerado um "kit de ferramentas", contendo componentes HTML, CSS e *JavaScript*, cujo uso proporciona rapidez e praticidade ao desenvolvimento. Este *framework* tem como principal objetivo auxiliar na criação de sites amigáveis e responsivos. O *framework* surgiu, em primeiro momento como uma tentativa de resolver o problema de falta de padronização visual nos sistemas Web desenvolvidos na equipe do *Twitter*, composta pelos engenheiros *Jacob Torton* e *Mark Otto* em 2010 (SANTIAGO, 2020).

Além das vantagens descritas anteriormente, Santiago (2020) acrescenta que ele oferece uma grande variedade de plug-ins, temas compatíveis com vários outros frameworks e possui suporte para todos os navegadores. A responsividade proporcionada pelo *Bootstrap* permite que os usuários possam acessar os sites em todos os tipos de dispositivos de forma otimizada e sem que informações sejam perdidas durante a navegação.

Na figura 26, vemos como importar os arquivos CSS e *JavaScript* do *Bootstrap* na versão mais atual e estável no momento de desenvolvimento do documento, enquanto a figura 27 exemplifica a visão e poder do *framework*, junto com códigos e arquivos personalizados para um site.

Figura 28 - Importando Bootstrap CSS e JS no HTML

```
<!-- Importando Bootstrap CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css">

<!-- Importando Bootstrap JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.js"></script>
```

Fonte: Autoria Própria, 2023.

Figura 29 - Exemplo de um site utilizando Bootstrap

Fonte: Autoria Própria, 2022.

2.12.2 Laravel

O *Laravel* é um framework baseado na linguagem *Hypertext Preprocessor* (PHP) utilizado para o desenvolvimento *web*, que possui uma arquitetura *Model View Controller* (MVC), que tem como o aspecto principal, o de auxiliar no desenvolvimento

de aplicações seguras e de alto desempenho de forma ágil e simplificada, com o código limpo (PELIZZA, 2018).

De acordo com Pelizza (2018), outra das características do *framework* é o incentivo de uso das boas práticas de programação e utilização de padrões específicos determinados para ele, tendo assim como outros presentes na área testes já definidos e estrutura com métodos auxiliares convenientes, que permitem testar expressamente suas aplicações.

Devido à grande quantidade de possibilidades que oferece, o *Laravel* utiliza o Composer como gerenciador das mais variadas dependências relacionadas a ele. O principal comando de execução é o *artisan*, que pode ser considerado o mais variado dentre os comandos utilizados em frameworks e é independente do gerenciador de dependências. Com o *Artisan*, os desenvolvedores podem criar arquivos de código-fonte, executar migrações de banco de dados, gerar modelos e controladores, gerar códigos para autenticação, entre muitas outras tarefas. Ele também permite que os desenvolvedores criem seus próprios comandos personalizados para executar tarefas específicas dentro de seus projetos *Laravel*, sendo uma ferramenta fundamental para aumentar a produtividade e eficiência no desenvolvimento de aplicações (GABARDO, 2017).

O *template* principal do *front-end* referente ao *framework* é o *Blade*, que, de acordo com o manual oficial do site, é um motor de *template* simples, mas poderoso. Ao contrário de alguns motores de *template* em PHP, ele não restringe o uso de código PHP puro neles, o que faz com que o *template* adicione uma sobrecarga mínima ao site (DOUGLAS, 2017).

Com relação ao MVC, o *Laravel* o utiliza da seguinte forma:

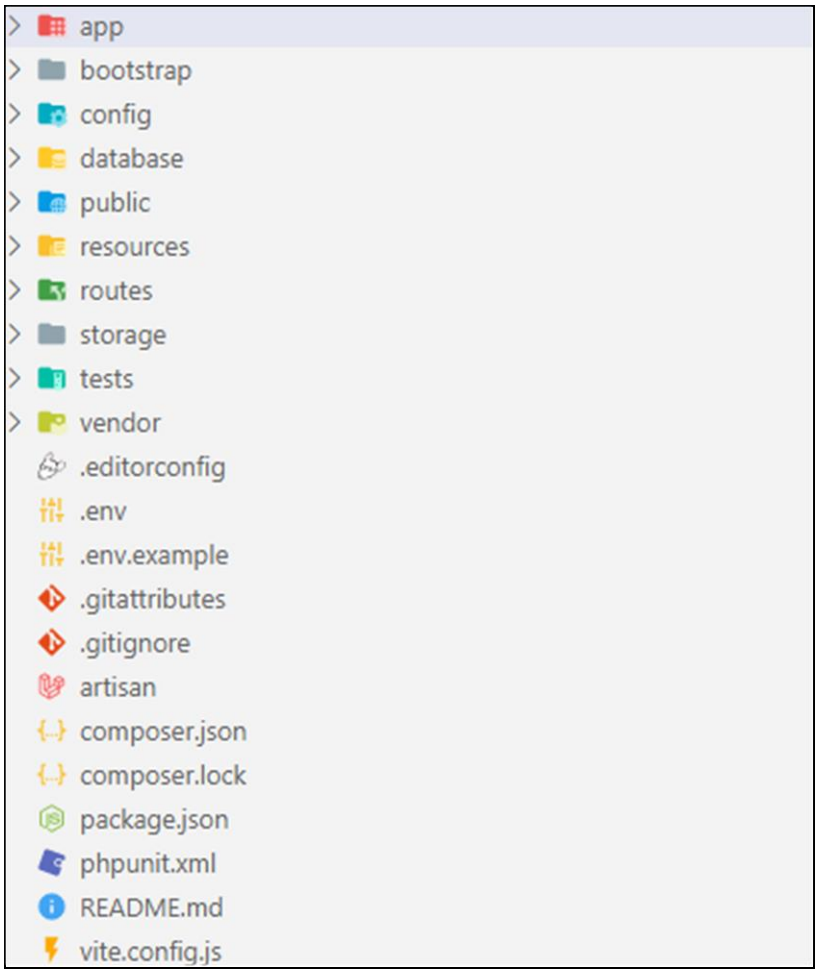
- A *View* é desenvolvida com base nos arquivos da pasta de mesmo nome, que fica em *resources*. É nela que fica os dados de Documento HTML das páginas do site, que se tem o auxílio de desenvolvimento do *template* citado anteriormente.
- A declaração de rotas (*routes*), que são basicamente os endereços possíveis de se acessar em uma *webpage*, são feitos pelo *web.php*, que fica na pasta de nome anterior. Nela, é possível você fazer declarar as rotas principais do site e retornar o *index.blade.php* (página) específico, ou também, em casos mais

complexos, chamar uma classe *Controller*, que servirá para retornar páginas específicas dependendo de uma condição implementada, sendo bem mais lógica e obrigatória em ser fazer, caso seu projeto tenha uma escalabilidade maior.

- Referente ao *Model*, não existe muita diferença com a parte teórica explicado no penúltimo subtópico, mas pode-se dizer que é um dos charmes principais do *Laravel*, a sua diversificada conexão com variados bancos de dados e a forma de controle, junto com suas classes *Eloquent*, *Provider*, *Factories*, *Seeders* e *Migrations*.

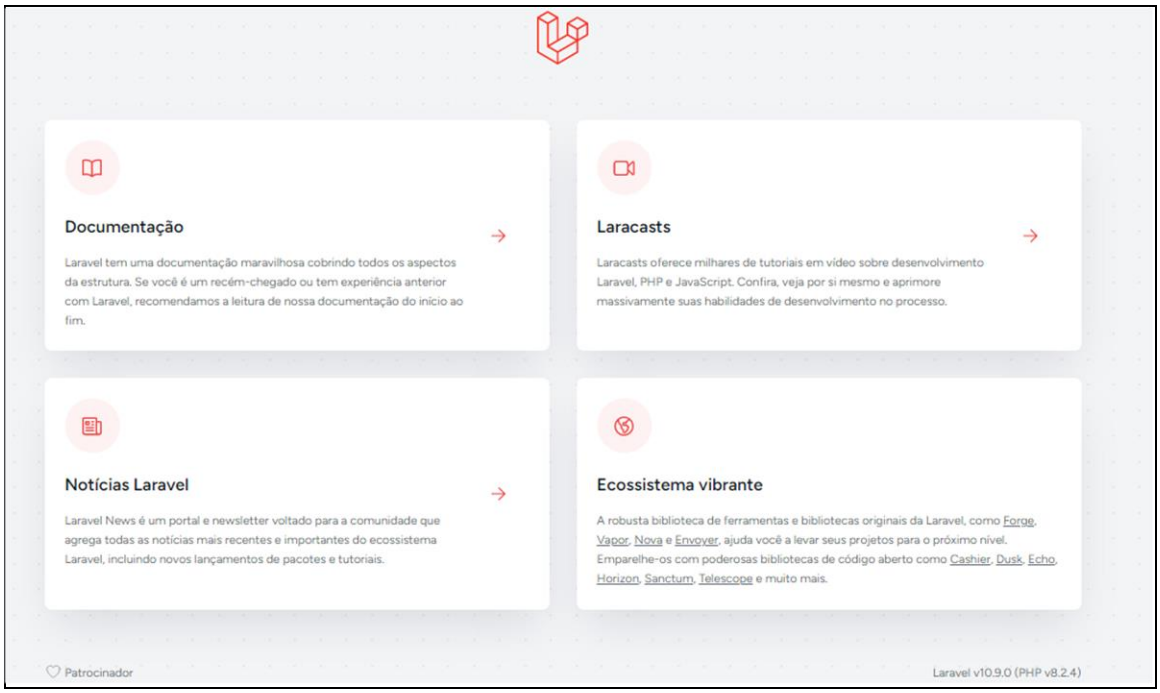
Na figura 30 vemos a estrutura padrão do *Laravel*, junto com o que aparece quando você utiliza o comando *php artisan serve* pela primeira vez na figura 31, sem alterar nada no projeto instalado completamente pelo *Composer*. O exemplo feito foi seguindo a versão mais recente do *Laravel*: a 10.x.

Figura 30 - Estrutura de um projeto Laravel



Fonte: Autoria Própria, 2023.

Figura 31 - Visão inicial de um projeto Laravel



Fonte: Autoria Própria, 2023.

2.13 Sistema de Banco de Dados NoSQL

O grande volume de dados gerado por aplicações *Web*, juntamente com os requisitos diferenciados destas aplicações, como a escalabilidade sob demanda e o elevado grau de disponibilidade, têm contribuído para o surgimento de novos paradigmas e tecnologias. As redes sociais, por exemplo, requerem o gerenciamento de grandes quantidades de dados não estruturados, os quais são gerados diariamente por milhões de usuários em busca do compartilhamento de informações, conhecimentos e interesses. Neste contexto, surgiu uma nova categoria de Banco de Dados, chamada *Not Only SQL* (*NoSQL*), que foi proposta com o objetivo de atender aos requisitos de gerenciamento de grandes volumes de dados, semiestruturados ou não estruturados, que necessitam de alta disponibilidade e escalabilidade (LÓSCIO, 2011).

Inicialmente, as propostas de bancos de dados não relacionais foram desenvolvidas por pequenas empresas e por comunidades de software livre. Tais soluções foram então agrupadas em um termo, *NoSQL*, que significa “não apenas SQL”. Este termo faz referência ao Sistema de Gerenciamento de Banco de Dados (SGBD) que não adotam o modelo relacional e são mais flexíveis referente aos pilares da Atomicidade, Consistência, Isolamento, Durabilidade (ACID) em um SGBD. Esta flexibilidade torna-

se necessária devido aos requisitos de alta escalabilidade necessários para gerenciar grandes quantidades de dados, bem como para garantir a alta disponibilidade dos mesmos, característica fundamental para as aplicações da *Web 2.0* (LÓSCIO, 2011).

Referente à taxonomia do *NoSQL*, segundo Diana (2010), existem 4 tipos em específico:

1. Bancos de dados orientados a documentos
2. Armazéns chave-valor
3. Bancos de dados de famílias de colunas
4. Bancos de dados de grafos

O banco de dados utilizado no projeto (*Firebase*), utiliza o primeiro, que resumidamente são coleções de atributos e valores, onde um atributo pode ser multivalorado, que em geral não possuem esquema. Ou seja, os documentos armazenados não precisam possuir estrutura em comum, além usarem um formato de armazenamento que suporte que uma estrutura seja embutida em outra, como o *JavaScript Object Notation* (JSON).

2.13.1 Firebase

Desenvolvido pela *Google*, o *Firebase* é um conjunto de produtos distribuído gratuitamente, com um limite de utilização, em que dentre esses produtos, existem serviços de hospedagem, armazenamento em nuvem e banco de dados (GOOGLE, 2021).

A ferramenta permite, com poucas linhas de código, adicionar o banco de dados em aplicações web, Android e iOS para que se conectem ao mesmo banco, sem requerer muitos conhecimentos sobre a infraestrutura do sistema (GOOGLE, 2021).

Um exemplo do que o Firebase tem a oferecer referente ao *BackEnd* é o *Firestore*, um banco de dados de documentos *NoSQL* que permite armazenar, sincronizar e consultar dados facilmente para seus apps para dispositivos móveis e da *Web*, em escala global, fornecido com *SDKs* para dispositivos móveis e *Web*, além de um conjunto abrangente de regras de segurança para que você possa acessar seu banco de dados sem precisar manter seu próprio servidor (GOOGLE, 2021).

Além desse produto, o *Firebase* também oferece diversas outras funcionalidades, incluindo autenticação, armazenamento em nuvem, banco de dados em tempo real, *cloud functions*, *hosting*, *analytics* e *messaging*. Entre as principais vantagens do *Firebase*, é possível destacar a facilidade de uso, a escalabilidade, a segurança e a flexibilidade que oferece, junto também com sua simplicidade e abrangência nas várias linguagens de programação (GOOGLE, 2021).

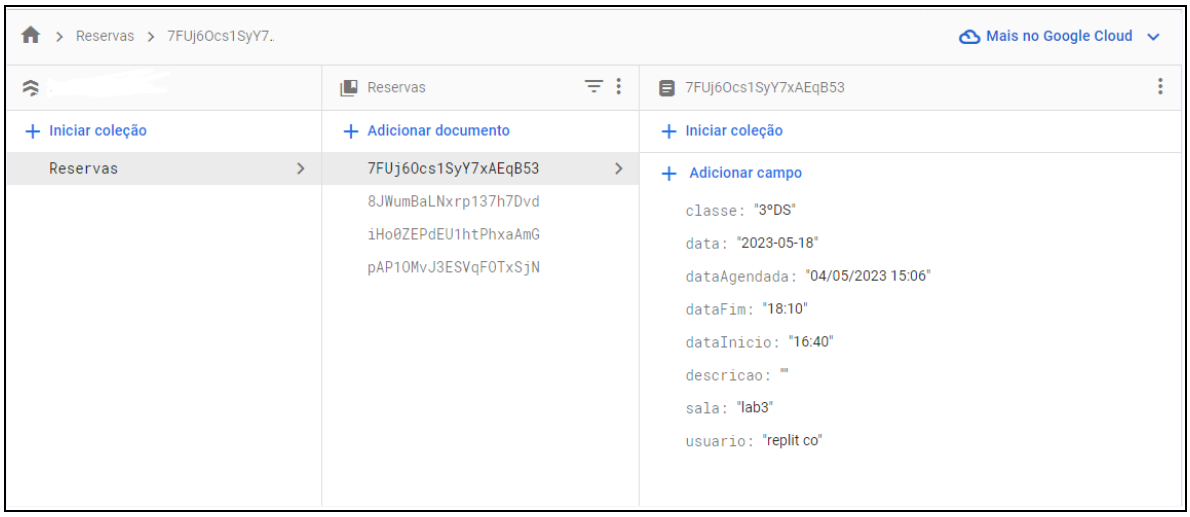
Na figura 32 vemos uma base do que é necessário para se fazer uma conexão do Javascript com o *Firebase* e o *Firestore*, enquanto na figura 33 temos uma demonstração do funcionamento no site do Projeto de Banco, sua divisão de documentos e coleções.

Figura 32 - Atributos Necessários para a conexão do Firebase com JS

```
const firebaseConfig = {
  apiKey: FIREBASE_API_KEY,
  authDomain:  FIREBASE_AUTH_DOMAIN,
  databaseURL:  FIREBASE_DATABASE_URL,
  projectId:  FIREBASE_PROJECT_ID,
  storageBucket:  FIREBASE_STORAGE_BUCKET,
  messagingSenderId:  FIREBASE_MESSAGING_SENDER_ID,
  appId:  FIREBASE_APP_ID,
  measurementId:  FIREBASE_MEASUREMENT_ID
};

const fbApp = initializeApp(firebaseConfig);
const fbDb = getFirestore(fbApp);
```

Fonte: Autoria Própria, 2023.

Figura 33 - Ambiente Cloud Firestore de um Banco do Firebase



Fonte: Autoria Própria, 2023.

2.14 UML

Segundo Bezerra (2007), *Unified Modeling Language* (UML) é uma linguagem de notação gráfica utilizada para representar e modelar sistemas de software de forma padronizada e clara. Através do uso de diagramas, a UML permite a representação visual dos componentes e estruturas de um sistema, além das suas interações e comportamentos. Cada diagrama é composto por elementos gráficos que se relacionam entre si para descrever diferentes aspectos do sistema, tornando mais fácil a compreensão por parte dos desenvolvedores e outras partes interessadas.

Sabendo do significado referente à UML, vamos para a estrutura utilizada nesse modelo e seu funcionamento. De acordo com Bezerra (2007), os diagramas da UML são categorizados em dois grupos principais: diagramas estruturais e diagramas comportamentais. Os diagramas estruturais são utilizados para especificar detalhes da estrutura do sistema, que são considerados a parte estática do sistema. Isso inclui elementos como classes, métodos, *interfaces*, *namespaces*, serviços, bem como a arquitetura geral do sistema e como os componentes devem ser instalados.

Sabendo disso agora podemos entender para que finalidade podemos utilizar os diagramas comportamentais para auxiliar no desenvolvimento de sistemas. Conforme Booch (2006), os diagramas comportamentais são utilizados para especificar como o sistema se comporta e interage com outros elementos externos a ele.

Esses diagramas abrangem a parte dinâmica do sistema, ou seja, como as funcionalidades devem funcionar, como um processo de negócio é tratado pelo

sistema, como os componentes estruturais trocam mensagens e como respondem às chamadas, entre outros aspectos relacionados ao comportamento do sistema. Em contrapartida, os diagramas estruturais especificam os detalhes da estrutura do sistema, enquanto os diagramas comportamentais são responsáveis por descrever o comportamento dinâmico do sistema (BOOCH, 2006).

2.14.1 Diagrama de Caso de Uso

O objetivo do diagrama de casos de uso é proporcionar uma visão abrangente e externa das funcionalidades que o sistema deve oferecer aos usuários, sem se aprofundar na forma como essas funcionalidades serão implementadas. Esse tipo de diagrama é comumente utilizado nas fases de elicitação e análise de requisitos do sistema, mas também pode ser consultado durante todo o processo de modelagem e servir como base para diversos outros diagramas (GUEDES, 2009).

Um ator é representado por um boneco, e é algo ou alguém que interage com o sistema, podendo ser um usuário humano ou um outro sistema computacional. Um caso de uso é representado por uma elipse e um rótulo com o nome do caso de uso. Define uma grande função do sistema (BOOCH, 2006).

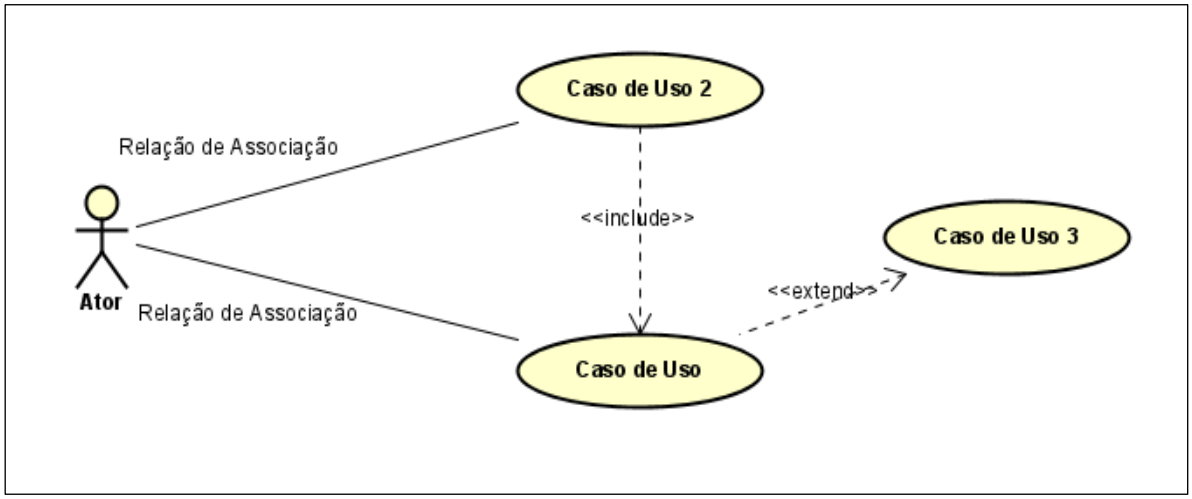
O *Include* é um relacionamento, que indica que quando um caso de uso for executado talvez um outro possa ser executado também, enquanto o *Extends*, também um relacionamento, em que, nesse caso, representa que para um caso de uso ser executado talvez dependa de um outro. No sistema de gerenciamento de salas, o usuário irá inserir todas as necessidades. Em seguida o sistema mostrara as salas disponíveis com as determinadas necessidades do usuário (JÚNIOR, 2020).

Itens utilizados para construção do diagrama de caso de uso:

- Elipses representam um Caso de Uso
- Bonecos representam os atores
- Setas não pontilhadas representam um relacionamento de associação
- Setas Pontilhadas com “*include*” escrito entre “<< >>” representam um relacionamento de inclusão.
- Setas Pontilhadas com “*extends*” escrito entre “<< >>” representam um relacionamento de extensão.

Na figura 34 um exemplo dos itens utilizados para construção do diagrama de caso de uso de forma visual.

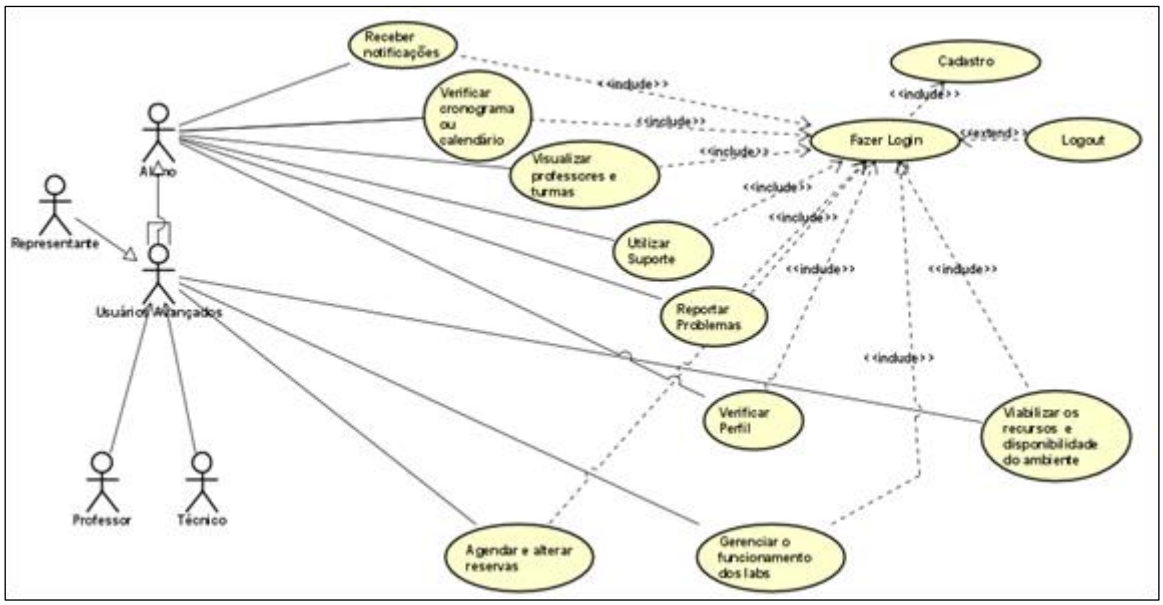
Figura 34 - Exemplo dos itens utilizados no Caso de Uso



Fonte: Autoria Própria, 2023.

Na figura 35 temos um exemplo na prática de um caso de uso, contendo uma estrutura completa e suas regras.

Figura 35 - Exemplo de Caso de Uso



Fonte: Autoria Própria, 2023.

2.14.2 Diagrama de Atividades

Diagramas de atividade ajudam na comunicação entre as pessoas das áreas de negócios e de desenvolvimento de uma organização para entender o mesmo processo e comportamento. Para criar um diagrama de atividade, é necessário um

conjunto de símbolos especiais, incluindo aqueles para dar partida, encerrar, fundir ou receber etapas no fluxo o qual abordaremos de forma mais aprofundada. (MURTA, 2015).

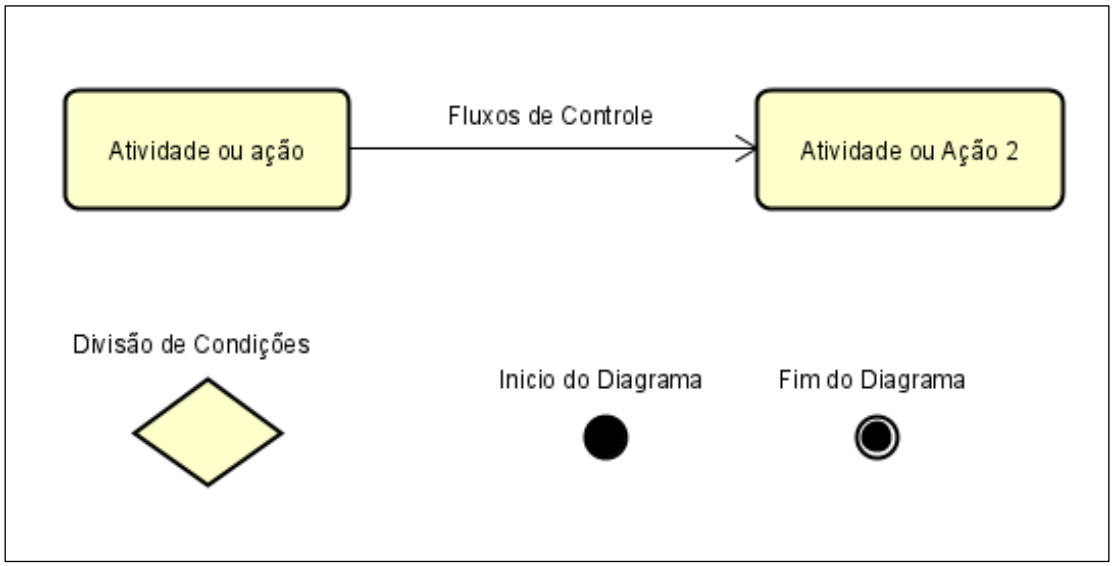
Usados para modelar os requisitos do negócio, criar uma visão de alto nível das funcionalidades de um sistema, analisar casos de uso e para vários outros fins.

Itens utilizados para construção do diagrama de atividade:

- Caixas representando atividades ou ações
- Linhas representando fluxos de controle
- Losango representando divisão de condições
- Círculos preenchidos representando o início do diagrama
- Círculos com bordas representando o fim do diagrama

Na figura 36 um exemplo dos itens utilizados para construção de um diagrama de atividade de forma visual.

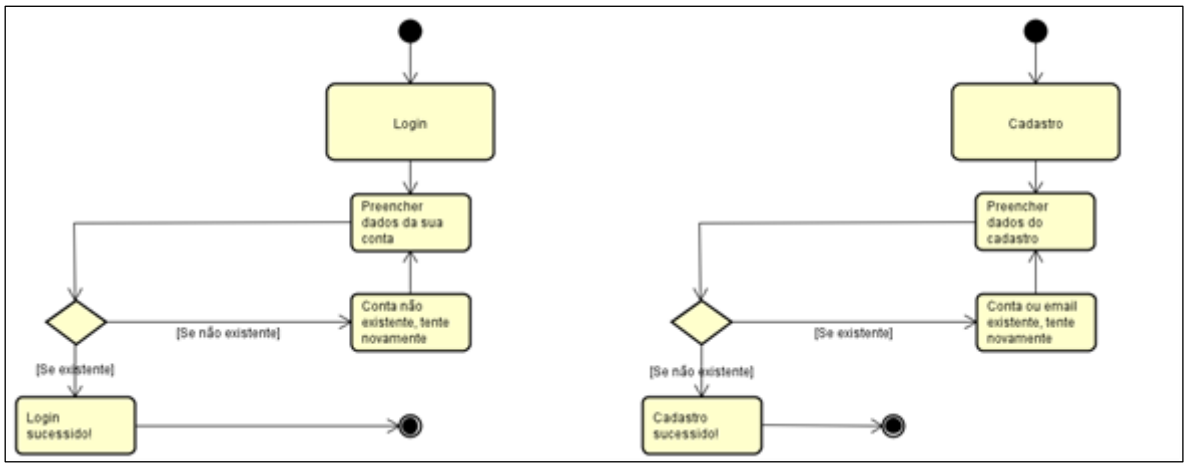
Figura 36 - Exemplo dos itens utilizados no Diagrama de Atividade



Fonte: Autoria Própria, 2023.

Na figura 37 temos um diagrama de atividade que representa as Funcionalidades de Login e Cadastro de um sistema.

Figura 37 - Diagrama de Atividade para o Login e Cadastro de Usuário



Fonte: Autoria Própria, 2023.

2.14.3 Diagrama de Classes

Os diagramas de classe são fundamentais para o processo de modelagem de objetos e modelam a estrutura estática de um sistema. Dependendo da complexidade de um sistema, é possível utilizar um único diagrama de classe para modelar um sistema inteiro ou vários diagramas de classe para modelar os componentes de um sistema.

Eles são as cópias do sistema ou subsistema. Você pode utilizar os diagramas de classe para modelar os objetos que compõem o sistema, para exibir os relacionamentos entre os objetos e para descrever o que esses objetos fazem e os serviços que eles fornecem. (JÚNIOR, 2020).

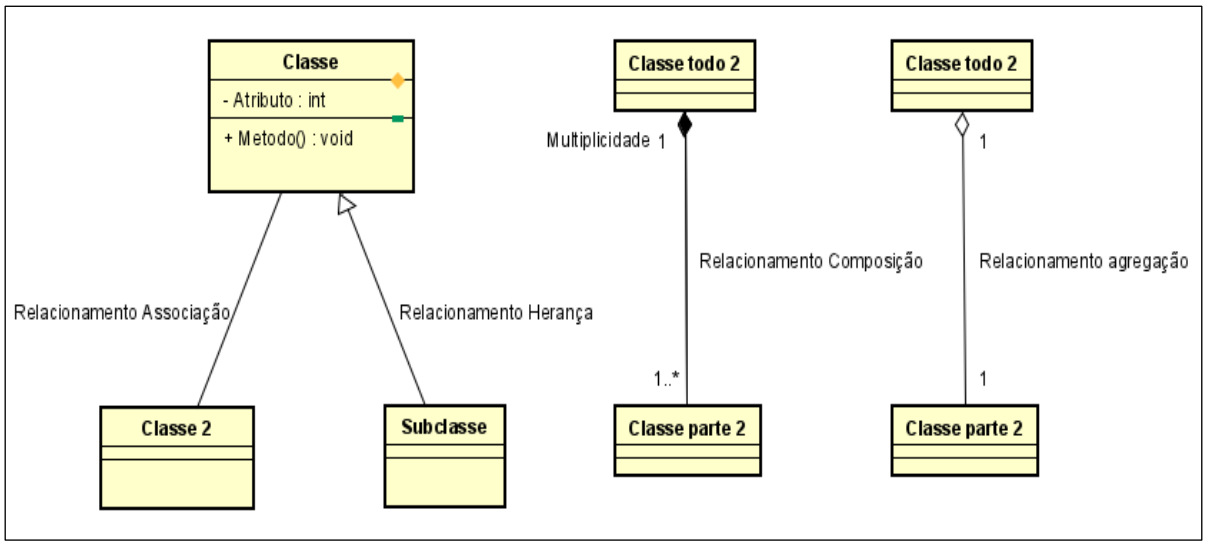
Além disso, são úteis em muitos estágios do design do sistema. No estágio de análise, um diagrama de classe pode ajudá-lo a compreender os requisitos do domínio do problema e a identificar seus componentes. Em um projeto de software orientado a objetos, os diagramas de classe criados durante os estágios iniciais do projeto contêm classes que normalmente são convertidas em classes e objetos de software reais quando você grava o código. Posteriormente, é possível refinar a análise e os modelos conceituais anteriores em diagramas de classe que mostrem as partes específicas do sistema, interfaces com o usuário, implementações lógicas e assim por diante. Os diagramas de classe tornam-se, então, uma captura instantânea que descreve exatamente como o sistema funciona, os relacionamentos entre os componentes do sistema em vários níveis e como planeja implementar esses componentes (DE ALMEIDA, 2006).

Os itens básicos utilizados para construção de um diagrama de classes são os seguintes:

- Classe: É representada por um retângulo dividido em três seções. A primeira seção contém o nome da classe, a segunda seção apresenta os atributos e a terceira seção exibe os métodos.
- Atributo: É representado dentro da seção de atributos da classe, geralmente listado como nome do atributo seguido pelo seu tipo de dado.
- Método: É representado dentro da seção de métodos da classe, normalmente listado como nome do método seguido pelos seus parâmetros (se houver) entre parênteses e, opcionalmente, o tipo de retorno do método.
- Associação: É representada por uma linha que conecta as classes envolvidas. A linha pode ter uma seta em uma das extremidades para indicar a direção do relacionamento. A multiplicidade (quantos objetos estão envolvidos na associação) é indicada perto das extremidades da linha.
- Herança: É representada por uma linha com uma seta sólida que aponta da subclasse para a superclasse. Às vezes, a linha pode ter um triângulo vazio no extremo da superclasse.
- Agregação: É representada por uma linha com um losango no extremo que aponta para a classe inteira (todo). A linha conecta a classe parte com a classe inteira.
- Composição: É representada por uma linha com um losango preenchido no extremo que aponta para a classe inteira (todo). A linha conecta a classe parte com a classe inteira.

Na figura 38 um exemplo dos itens utilizados para construção de um diagrama de classes de forma visual.

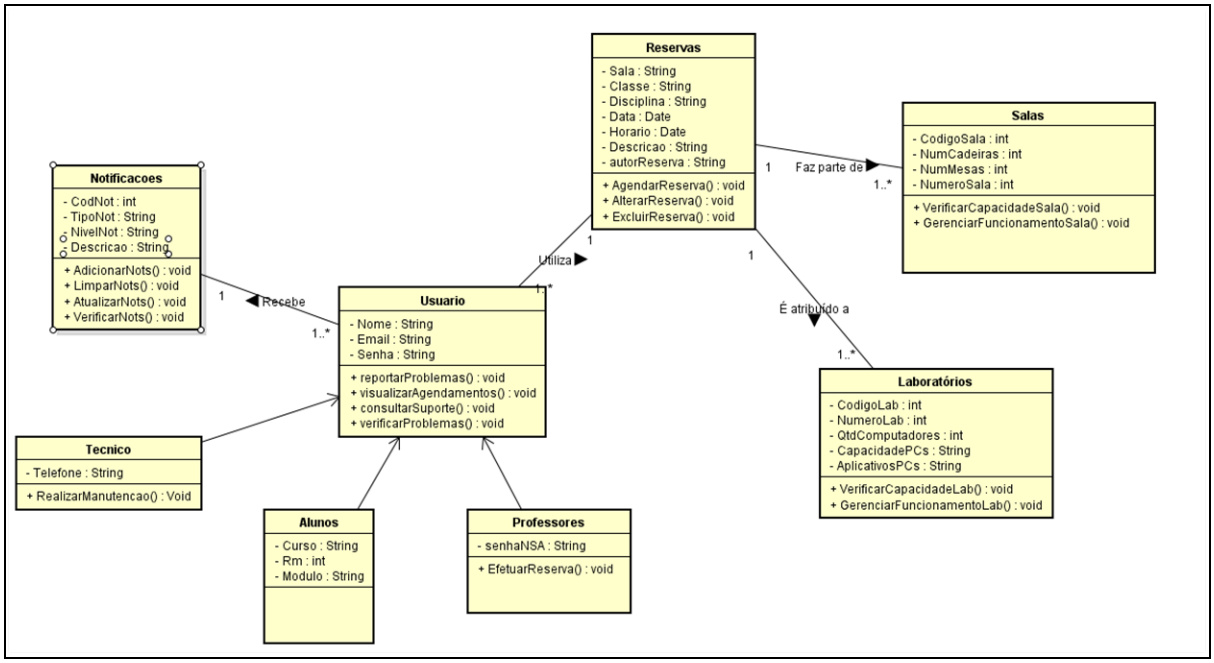
Figura 38 - Exemplo dos itens utilizados no Diagrama de Classe



Fonte: Autoria Própria, 2023.

Na figura 39 apresenta um exemplo de um diagrama de classes completo de um sistema onde possui todos os itens citados acima.

Figura 39 - Diagrama de Classes do projeto



Fonte: Autoria Própria, 2023.

2.14.4 Diagrama de Sequência

De acordo com Guedes (2009), o diagrama de sequência é uma ferramenta comportamental que representa a ordem temporal em que as informações são transmitidas entre os objetos envolvidos em um processo específico. Geralmente, é

construído a partir de um caso de uso definido no diagrama de caso de uso e utiliza o diagrama de classes para identificar os objetos das classes envolvidas no processo.

Agora vamos adiante sobre o funcionamento e as vantagens do diagrama de sequência, segundo Fowler (2014), o diagrama de sequência é composto por objetos e mensagens, que são representados graficamente e possuem uma notação padronizada, o que torna sua compreensão bastante intuitiva e dispensa a necessidade de muitas explicações adicionais. Além disso, ele permite visualizar de forma clara e detalhada como os objetos interagem e trocam informações ao longo do processo em questão.

Vamos pegar um exemplo para melhorar o entendimento e fixação do tópico diagrama de sequência. Tenhamos como base o exemplo disponibilizado por Fowler (2014), que consiste em pedido de um cliente em uma loja virtual. Imagine que um cliente faça um pedido em uma loja virtual e deseje calcular o preço total do seu pedido, considerando os descontos aplicáveis. Nesse caso, o diagrama de sequência poderia mostrar a interação entre os objetos envolvidos nesse processo, como o pedido, os produtos e o cliente. O pedido, por sua vez, precisaria acessar as informações dos produtos para calcular o preço total do pedido, levando em conta possíveis descontos aplicáveis. O diagrama de sequência seria útil para representar visualmente essa sequência de ações, tornando mais fácil a compreensão do processo como um todo.

Os itens básicos utilizados para construção de um diagrama de sequência são os seguintes:

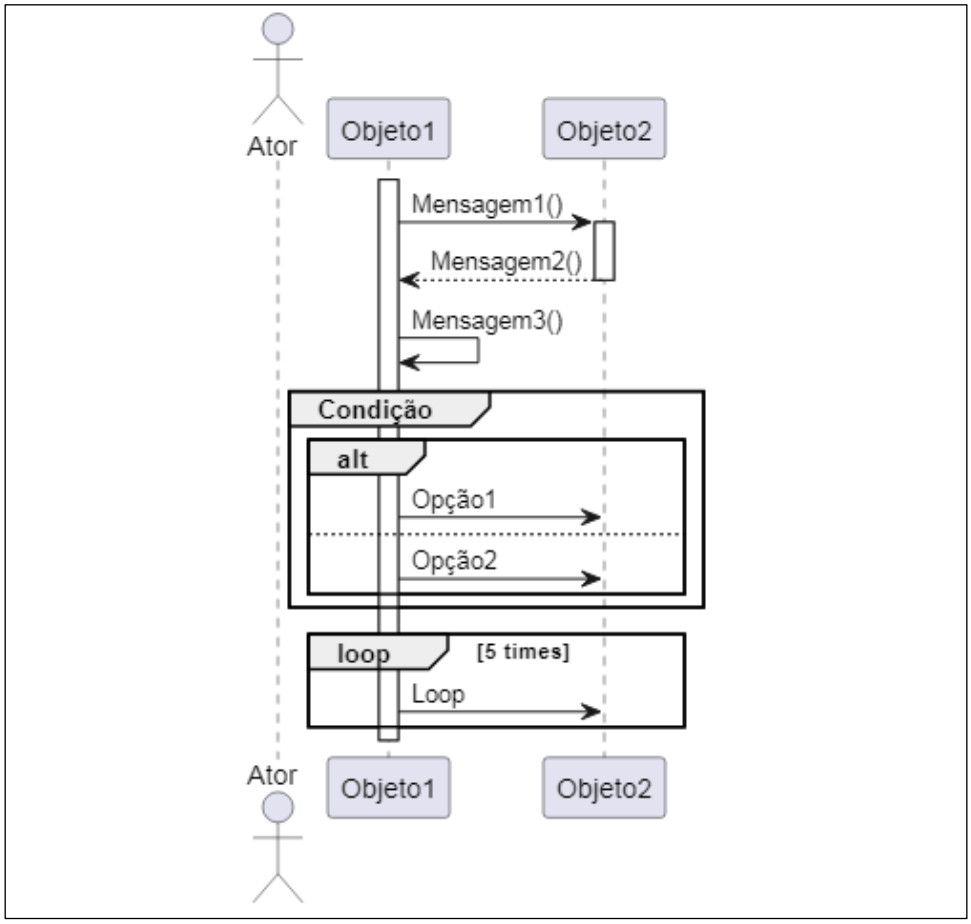
- Ator: É representado por um ícone de pessoa ou algum outro símbolo relevante, colocado no topo do diagrama, indicando a entidade externa que interage com o sistema.
- Objeto: É representado por um retângulo com o nome do objeto dentro dele, posicionado ao longo da linha de vida correspondente.
- Linha de Vida: É uma linha vertical que se estende de cima para baixo, representando a existência temporal de um objeto durante a interação. A linha de vida é associada a um objeto específico e ajuda a indicar quando o objeto está ativo.
- Mensagem: É representada por uma seta direcionada de um objeto para outro, acompanhada pelo nome da mensagem. As mensagens podem ser síncronas

(chamadas de métodos bloqueantes) ou assíncronas (chamadas de métodos não bloqueantes). Elas indicam a comunicação entre os objetos.

- Mensagem de Retorno: É uma resposta ou retorno de uma mensagem anteriormente enviada. É representada por uma seta pontilhada com uma linha de vida tracejada que volta para o objeto que originou a mensagem.
- Fragmentos (Fragmentos de Interação): São representados por retângulos com diferentes notações para indicar a lógica condicional, repetição ou paralelismo no diagrama de sequência. Por exemplo, o fragmento de opção é representado por um retângulo com uma barra diagonal e o fragmento de loop é representado por uma linha tracejada em volta de uma sequência repetitiva.
- Tempo: Pode ser representado por uma linha horizontal ou uma nota de tempo, que indica restrições de tempo, duração ou eventos específicos no diagrama.

Na figura 40 um exemplo dos itens utilizados para construção de um diagrama de sequencias de forma visual.

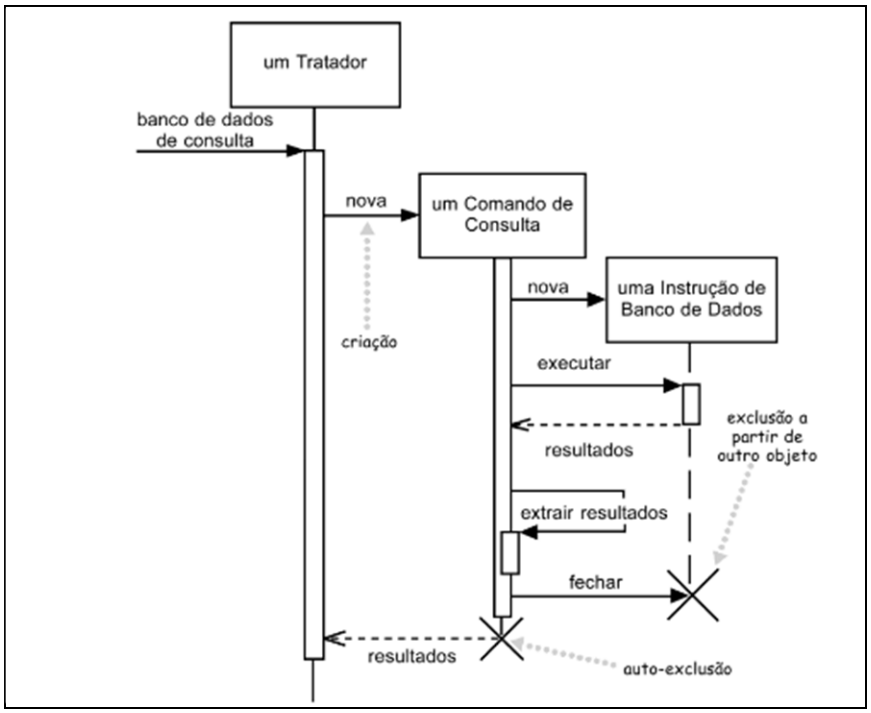
Figura 40 - Diagrama de Sequências de forma visual



Fonte: Autoria Própria, 2023.

Na figura 41 temos um exemplo de diagrama de sequência completo que representa a funcionalidade de criação e remoção de participantes.

Figura 41 - Diagrama de Sequência para criação e remoção de participantes



Fonte: Fowler (2014)

2.14.5 Diagrama de Objetos

Os diagramas de objetos têm um papel fundamental na UML (Unified Modeling Language), pois proporcionam uma representação visual das instâncias em um sistema, juntamente com os relacionamentos existentes entre elas. Esses diagramas capturam uma imagem instantânea das instâncias e seus estados em um momento específico (GUDWIN, 2014).

Um diagrama de objetos é um tipo de diagrama estrutural na UML que ilustra as instâncias dos classificadores presentes nos modelos. Sua notação é semelhante à dos diagramas de classes, mas, ao contrário destes, os diagramas de objetos mostram instâncias específicas dos classificadores, bem como os links que as conectam em um determinado instante (JÚNIOR, 2020).

A criação de um diagrama de objetos envolve a instanciação dos classificadores nos diagramas de classe, implementação, componente e caso de uso. Essa abordagem permite visualizar e analisar as instâncias e seus relacionamentos em um contexto

específico, proporcionando uma compreensão mais detalhada do comportamento do sistema (BEZERRA, 2007).

Os componentes-chave presentes em um diagrama de objetos são os seguintes:

1. Objetos: Representam as instâncias das classes em um sistema. Cada objeto é uma entidade individual com seu próprio estado e comportamento.

2. Títulos de classe: Refletem os atributos específicos de uma determinada classe. Esses títulos mostram as características particulares que cada objeto instanciado dessa classe possui.

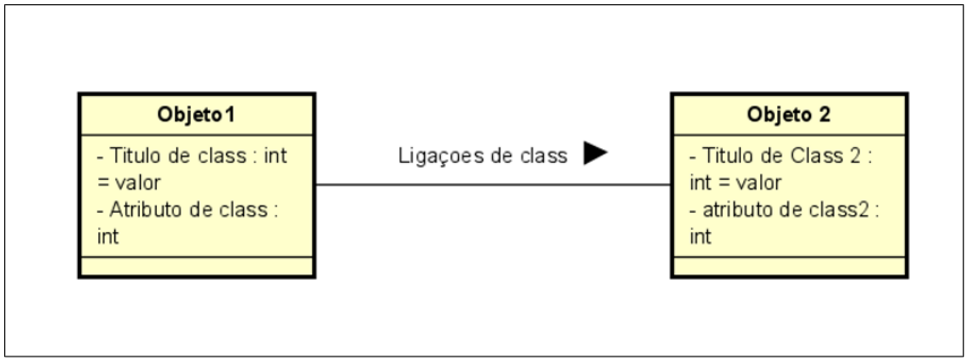
3. Atributos de classe: São as características dos objetos instanciados a partir de uma classe específica. Esses atributos são herdados pela classe e podem ter valores diferentes em cada instância.

4. Ligações: São linhas que conectam os objetos, representando os relacionamentos hierárquicos entre eles. Essas ligações podem mostrar associações, dependências ou outros tipos de relações entre os objetos.

Ao utilizar esses componentes em conjunto, um diagrama de objetos oferece uma visão detalhada das instâncias e seus relacionamentos, auxiliando na compreensão e modelagem do sistema.

Na figura 42 um exemplo dos itens utilizados para construção de um diagrama de objetos de forma visual.

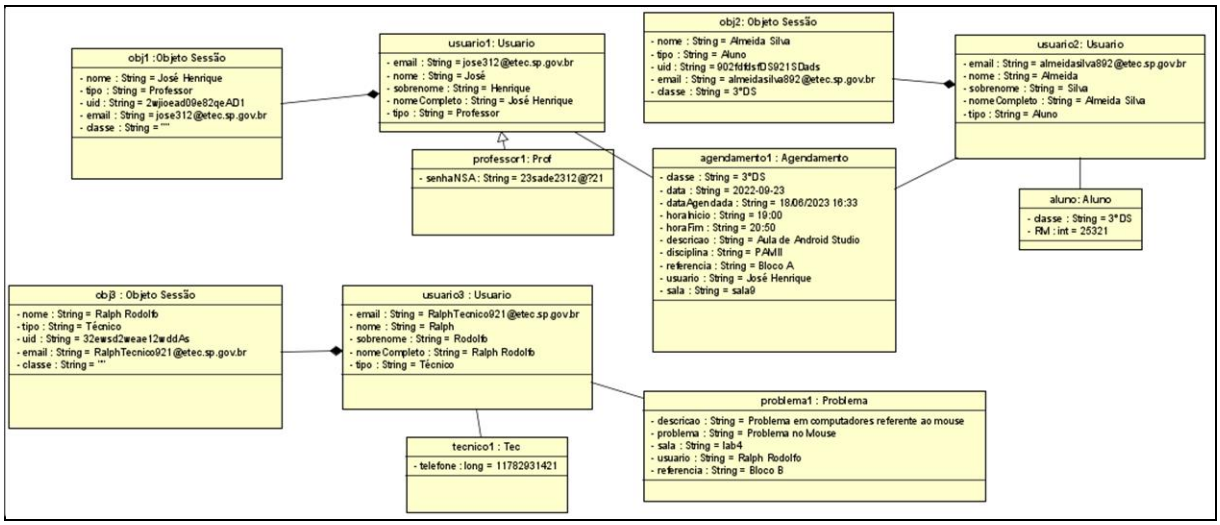
Figura 42 - Diagrama de Objetos de forma visual



Fonte: Autoria Própria, 2023.

Na figura 43, podemos observar um diagrama de objetos que representa os componentes do sistema de agendamento

Figura 43 - Componentes do Sistema



Fonte: Autoria Própria, 2023

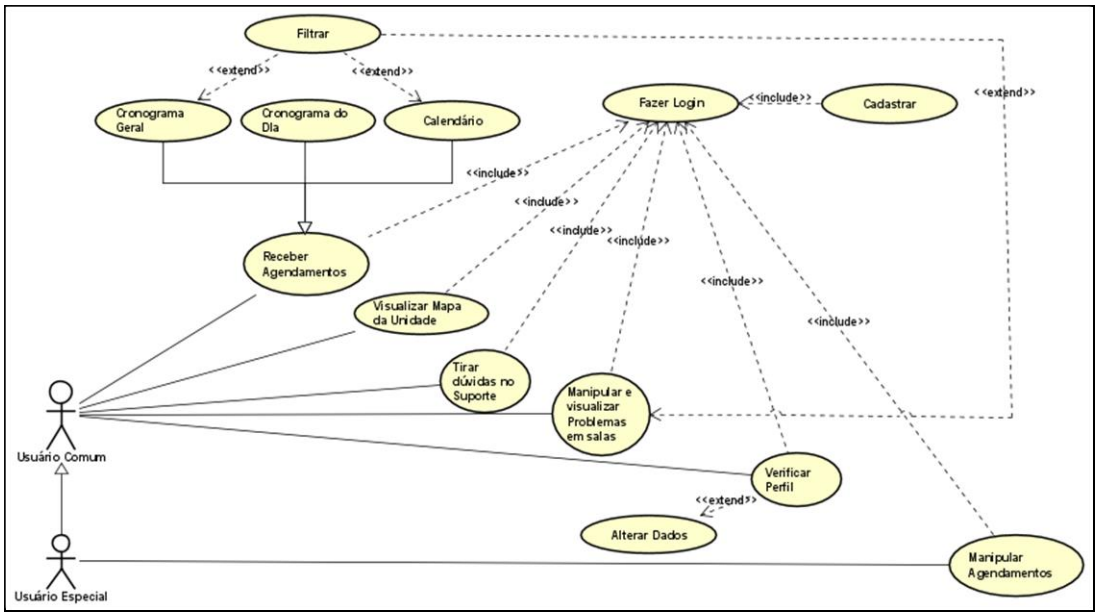
3 DESENVOLVIMENTO

Neste capítulo, abordaremos a construção e desenvolvimento do Sistema de Comunicação e Gerenciamento de Ambientes (SCGA). Apresentaremos os diagramas elaborados utilizando a UML como base, além de fornecer detalhes sobre as funcionalidades da aplicação. Também incluiremos a ilustração das principais telas do sistema, oferecendo uma visão geral da interface.

3.1 Diagrama de Caso de Uso

O diagrama de caso de uso apresentado na figura abaixo ilustra a interação de diferentes tipos de usuários com as funcionalidades do sistema. Essas funcionalidades incluem a possibilidade de fazer cadastro, receber e manipular agendamentos, visualizar mapa da unidade, entre outras aplicações.

Figura 44 - Diagrama de Caso de Uso Do Sistema SCGA



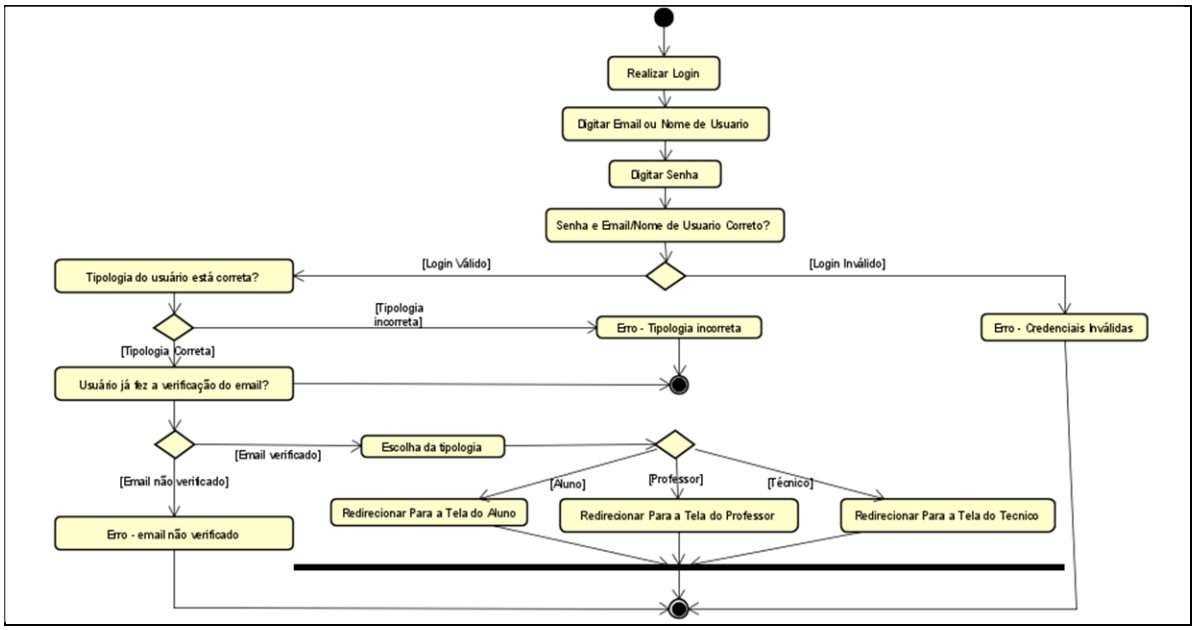
Fonte: Autoria Própria, 2023.

3.2 Diagrama de Atividade

Para demonstrar de forma detalhada o fluxo de cada funcionalidade do sistema foi utilizado o diagrama de atividade, esses diagramas foram divididos em sete partes, sendo elas: efetuar login, efetuar cadastro, verificação de perfil, visualizar agendamentos, reservar salas, alterar reserva e reportar problemas.

A figura 45 demonstra o fluxo para o usuário efetuar o login no sistema verificando algumas informações, primeiramente ele irá verificar se o usuário e senha que foi digitado está correto e de acordo com o cadastro, após revisar essas informações ele irá verificar se o usuário já verificou o Email cadastrado, se sim ele irá examinar se a topologia aluno, técnico ou professor selecionada e se estiver de acordo com o cadastro o login será efetuado.

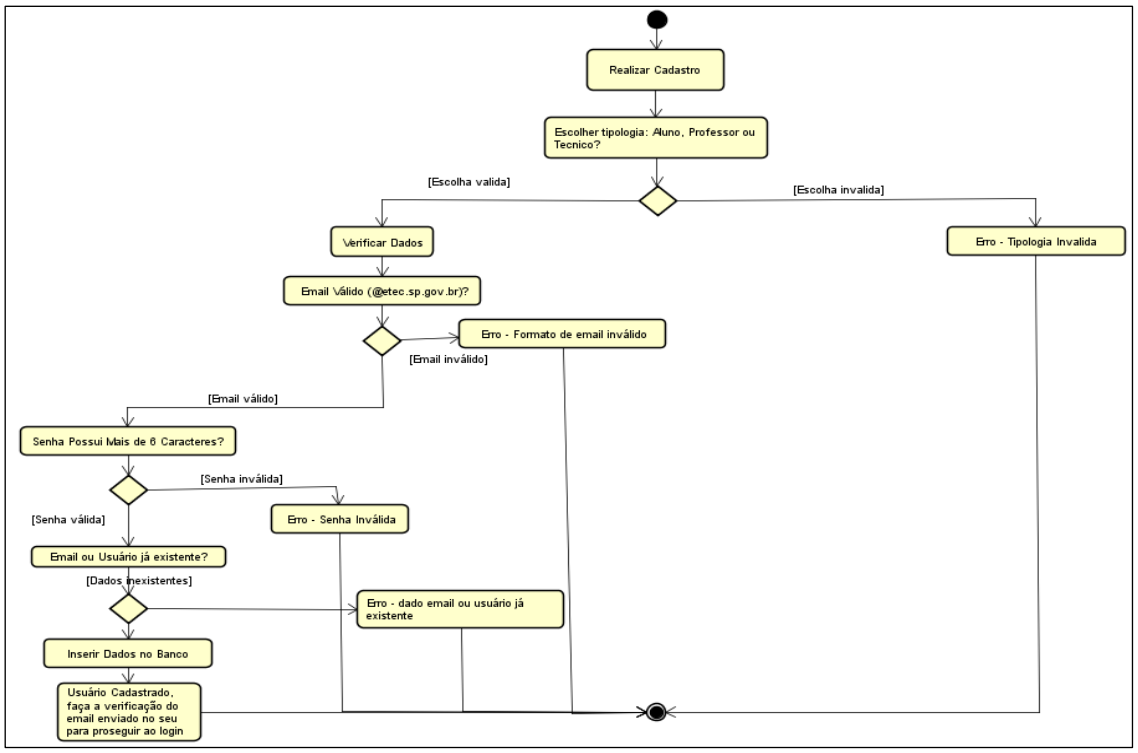
Figura 45 - Diagrama de Atividade do Login



Fonte: Autoria Própria, 2023.

A Figura 46 representa o fluxo do usuário durante o processo de cadastro no sistema. O usuário inicia o processo selecionando a tipologia na qual deseja se cadastrar, podendo escolher entre aluno, técnico ou professor. O sistema, então, verifica se a escolha do usuário é válida, ou seja, se está de acordo com as opções disponíveis para cadastro. Em seguida, são realizadas verificações de validação do Email e da senha digitados pelo usuário. O Email deve seguir o padrão “etec.sp.gov.br”, e a senha precisa ter, no mínimo, 6 caracteres por exemplo. Caso todas as informações estejam corretas, o usuário é cadastrado com sucesso no sistema.

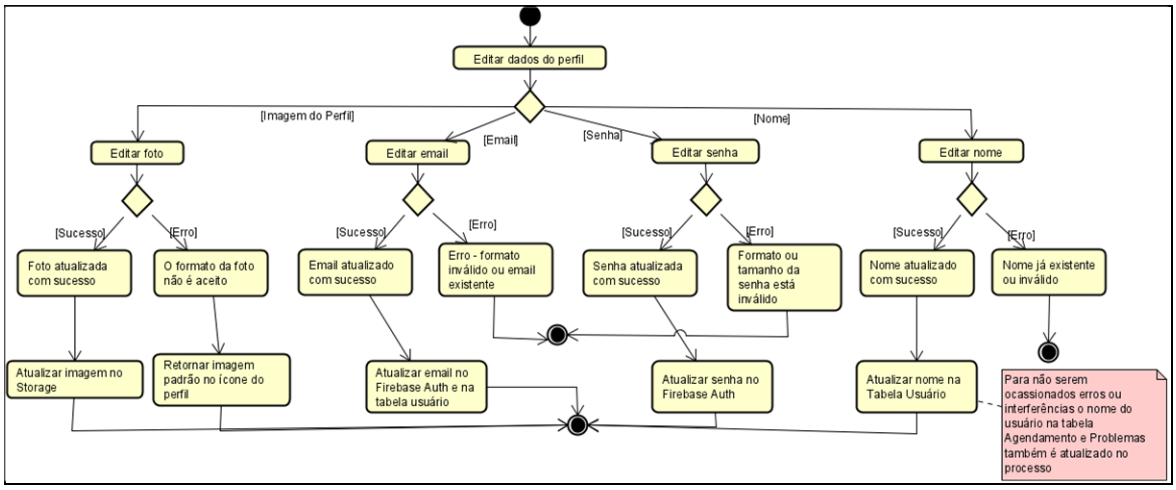
Figura 46 - Diagrama de Atividade do Cadastro



Fonte: Autoria Própria, 2023.

A Figura 47 ilustra o fluxo e a interação do usuário com o seu perfil. Ao optar por editar a foto de perfil, o sistema verifica se a foto está nos formatos aceitos, como PNG, JPEG, entre outros. Caso a foto não esteja de acordo com esses formatos, a edição não será realizada. Em seguida, se o usuário desejar editar o seu *e-mail*, o sistema verifica se o *e-mail* fornecido é institucional ou se o mesmo já não existe. Caso contrário, a edição não ocorrerá e será exibida uma mensagem de erro. Além disso, ao editar a senha, o sistema analisa se a senha atende ao formato correto e o tamanho. Caso a senha não cumpra os requisitos, será exibida uma mensagem de erro e por fim se o usuário desejar editar o nome será examinado se nome já existe ou se ele é válido para um nome, se não será possível editar o nome. Somente quando as informações que irão ser editadas estiverem corretas, o usuário poderá salvar as alterações realizadas.

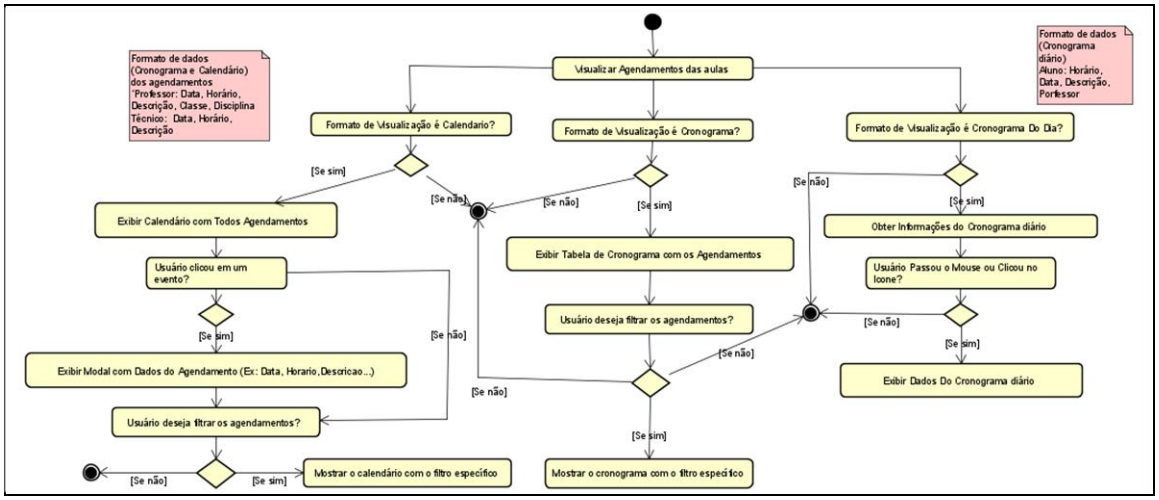
Figura 47 - Diagrama de Atividades para o Perfil do Usuário



Fonte: Autoria Própria, 2023.

A Figura 48 mostra o fluxo de visualização dos agendamentos das aulas, oferecendo ao usuário três opções: calendário mensal, cronograma e calendário diário na tela inicial. Ao selecionar o calendário mensal, o usuário pode visualizar todos os agendamentos do mês e obter detalhes clicando em um agendamento específico. No cronograma, todos os agendamentos são exibidos em uma tabela, permitindo ao usuário filtrar por agendamentos gerais ou específicos. No cronograma diário, as informações dos agendamentos são obtidas de acordo com a sua tipologia e os dados que batem com o dia, sendo exibidas ao passar o mouse sobre o ícone dele. Essas opções proporcionam ao usuário diferentes formas de visualizar e interagir com os agendamentos, permitindo uma melhor organização e acesso rápido às informações desejadas.

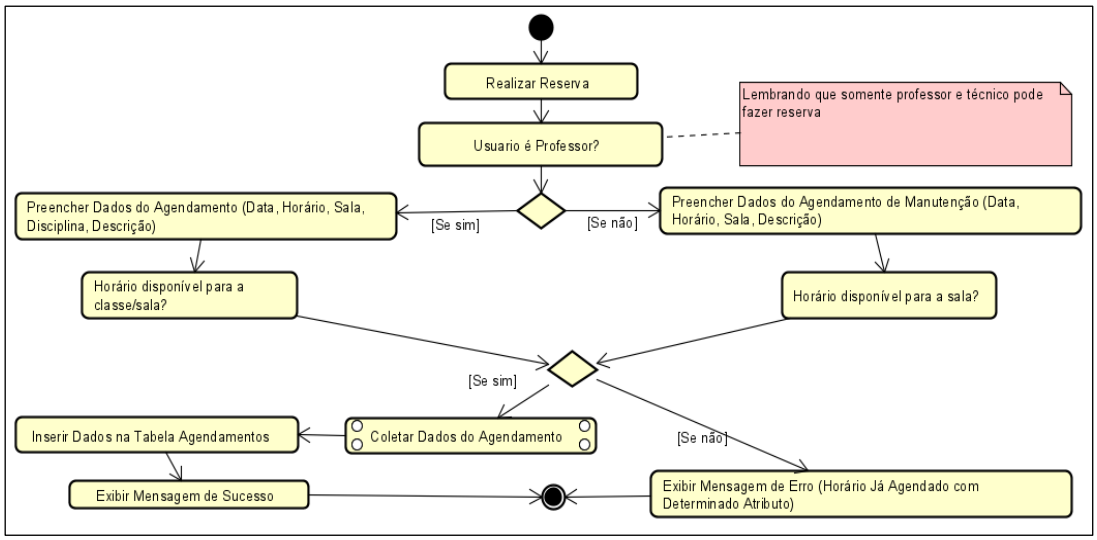
Figura 48 - Diagrama de atividade para visualização dos agendamentos das aulas



Fonte: Autoria Própria, 2023.

Na Figura 49, é mostrado o fluxo de atividades para a funcionalidade de reserva de salas. De início o sistema diferencia o tipo de usuário (professor ou técnico), pois as reservas são diferentes para cada categoria. O usuário preenche os dados do agendamento de acordo com sua categoria, e o sistema verifica a disponibilidade da sala. Se o horário estiver disponível, os dados do agendamento são inseridos na tabela e exibida uma mensagem de sucesso. Caso contrário, uma mensagem de erro é mostrada. Esse fluxo de atividades permite aos usuários realizar reservas adequadas, garantindo a disponibilidade das salas e proporcionando uma experiência eficiente.

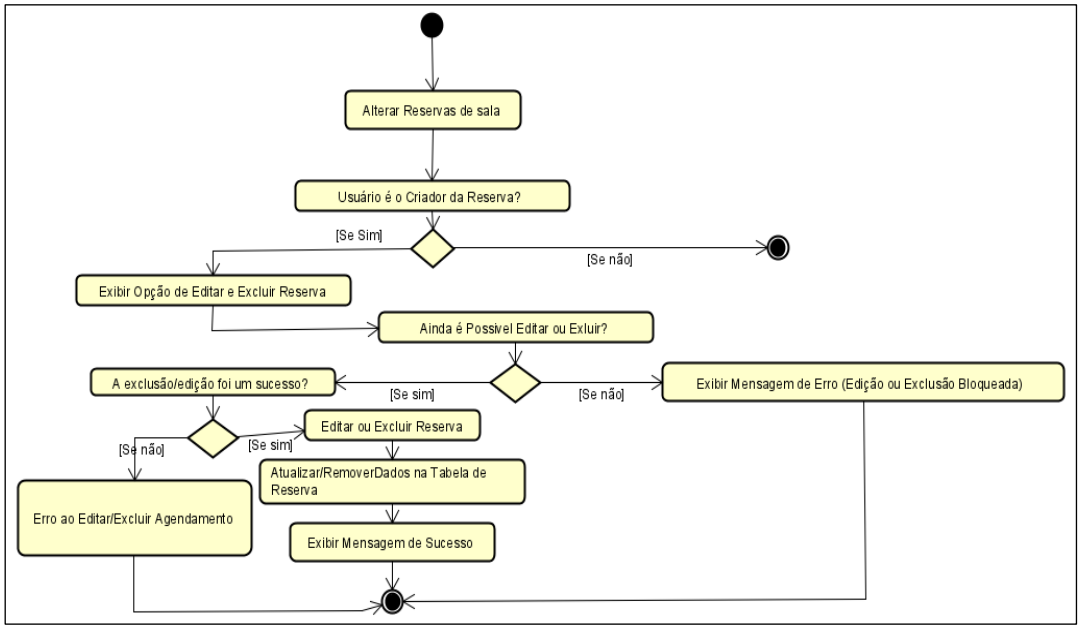
Figura 49 - Diagrama de atividade para realizar reserva de uma sala



Fonte: Autoria Própria, 2023.

A Figura 50 ilustra o fluxo para alterar uma reserva de sala no sistema. O usuário precisa ser o criador da reserva para realizar a modificação. Ao selecionar a opção de editar ou excluir a reserva, o sistema verifica se a ação foi executada com sucesso. Em caso positivo, os dados são atualizados na tabela e exibida uma mensagem de sucesso. Esse processo permite ao usuário gerenciar suas reservas de sala de forma eficiente e confiável, garantindo a integridade dos dados e proporcionando uma experiência satisfatória.

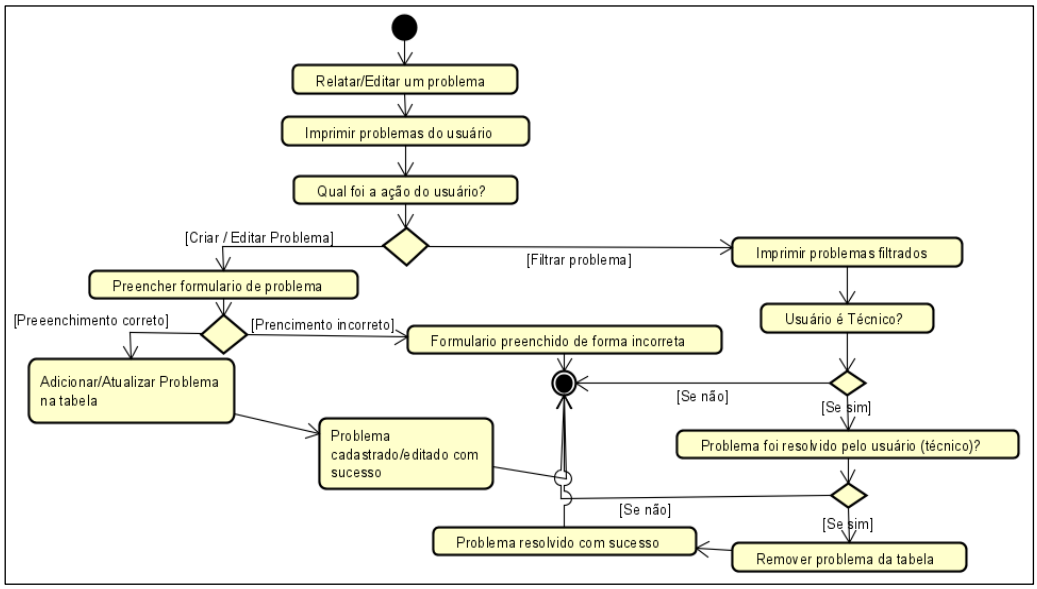
Figura 50 - Diagrama de atividade para alterar reserva de sala



Fonte: Autoria Própria, 2023.

A Figura 51 ilustra a funcionalidade de reportar e editar problemas por meio de um fluxo de etapas. Ao iniciar a atividade, todos os problemas são exibidos. O usuário pode escolher entre editar/criar um problema ou filtrar os problemas. Ao editar/criar um problema e preencher o formulário corretamente, o problema é registrado com sucesso. Ao optar por filtrar os problemas, os problemas filtrados são exibidos. Se o usuário for um técnico e o problema estiver resolvido, ele é removido da tabela e exibida uma mensagem de resolução. Esse fluxo permite reportar e editar problemas, filtrar informações e oferece uma funcionalidade específica para técnicos resolverem problemas, melhorando a experiência geral do usuário.

Figura 51 - Diagrama de atividades de reportar e editar problemas em ambientes



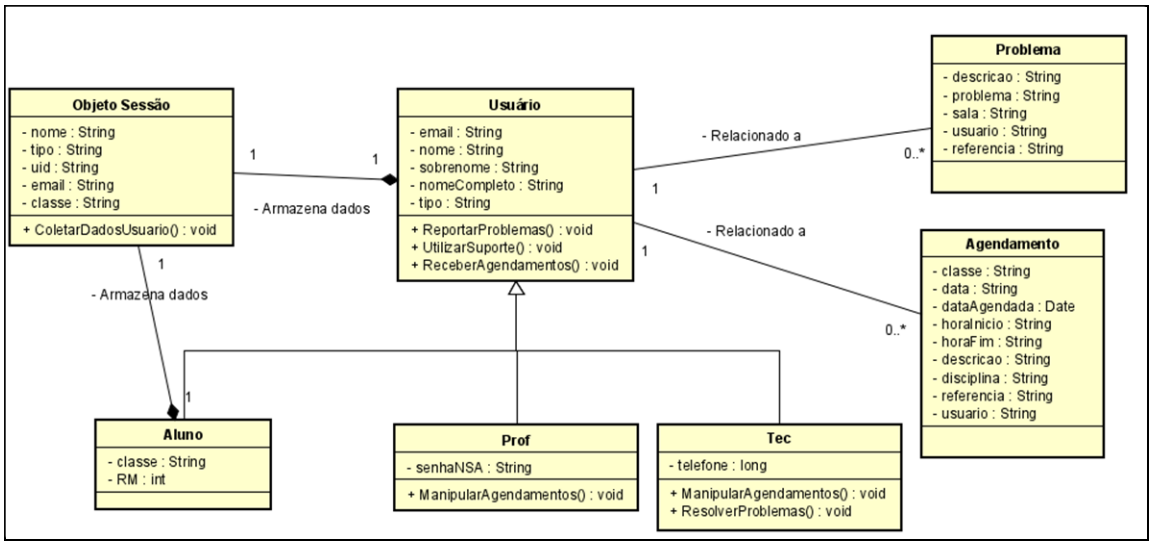
Fonte: Autoria Própria, 2023.

3.3 Diagrama de Classe

Para o desenvolvimento de sistemas de software mais abrangentes, é necessário adotar uma abordagem estruturada e organizada, a fim de garantir a eficiência, escalabilidade e facilidade de manutenção do sistema. Nesse contexto, os diagramas de classes desempenham um papel fundamental na engenharia de software orientada a objetos. Em nosso trabalho, utilizamos esses diagramas para identificar os atributos e funções necessários em nosso sistema por meio das classes representadas por figuras retangulares. Além disso, os diagramas de classes nos permitem entender as ações específicas que cada tipo de usuário pode realizar.

A figura 52 representa as classes que o sistema irá possuir, as relações dessas classes umas com as outras (como elas se interligam, como uma se conecta com a outra) e os atributos e métodos (funções) que cada classe possui por exemplo a classe usuário: possui os atributos nome, Email, senha e os métodos reportar problemas, visualizar agendamentos, consultar suporte e verificar problemas.

Figura 52 - Diagrama de Classes do Sistema



Fonte: Autoria Própria, 2023.

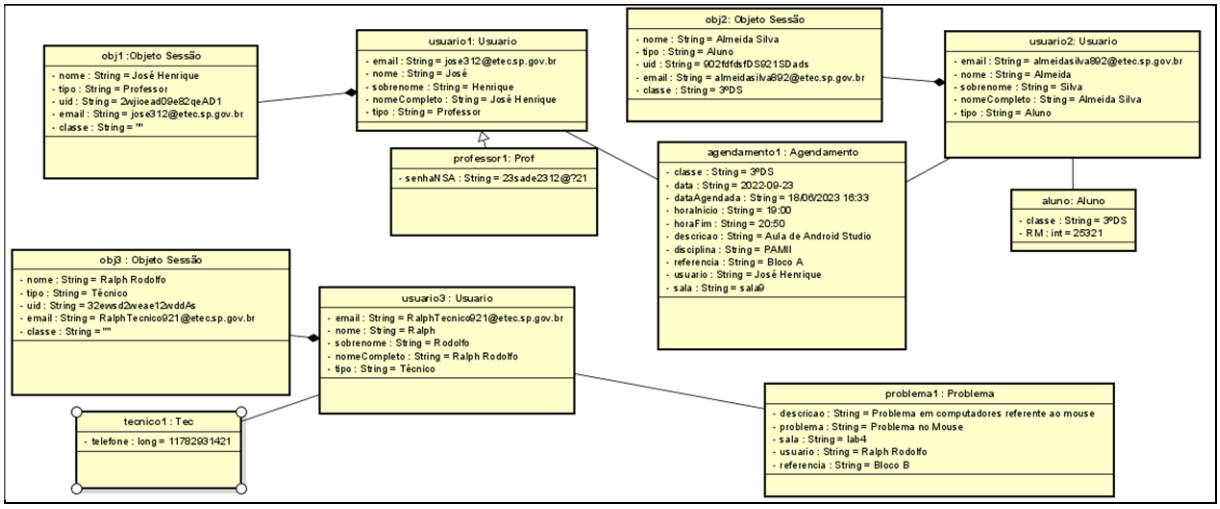
3.4 Diagrama de Objetos

O Diagrama de Objetos é uma das várias ferramentas disponíveis na UML (Unified Modeling Language) para modelar sistemas orientados a objetos. Ele é usado para representar uma visão estática e estrutural do sistema, mostrando os objetos que compõem o sistema, suas propriedades e relacionamentos. Das principais utilidades desse Diagrama pode-se citar a representação dos objetos dos sistemas e suas associações e identificá-los, modelar relações entre eles, a compreensão da estrutura dos dados e estados de objetos

No geral, o Diagrama de Objetos fornece uma representação visual poderosa para modelar a estrutura estática de um sistema orientado a objetos, ajudando a entender e comunicar sua arquitetura, componentes e interações entre os objetos.

Na figura 53 é representado os possíveis objetos e relações, como a conexão do objeto usuário com suas tipologias/generalizações (Aluno, Professor, Técnico), o objeto sessão (que armazena os dados do navegador do usuário necessários para relações, controle de dados e personalização) e o Agendamento e Problema.

Figura 53 - Diagrama de Objetos do Sistemas



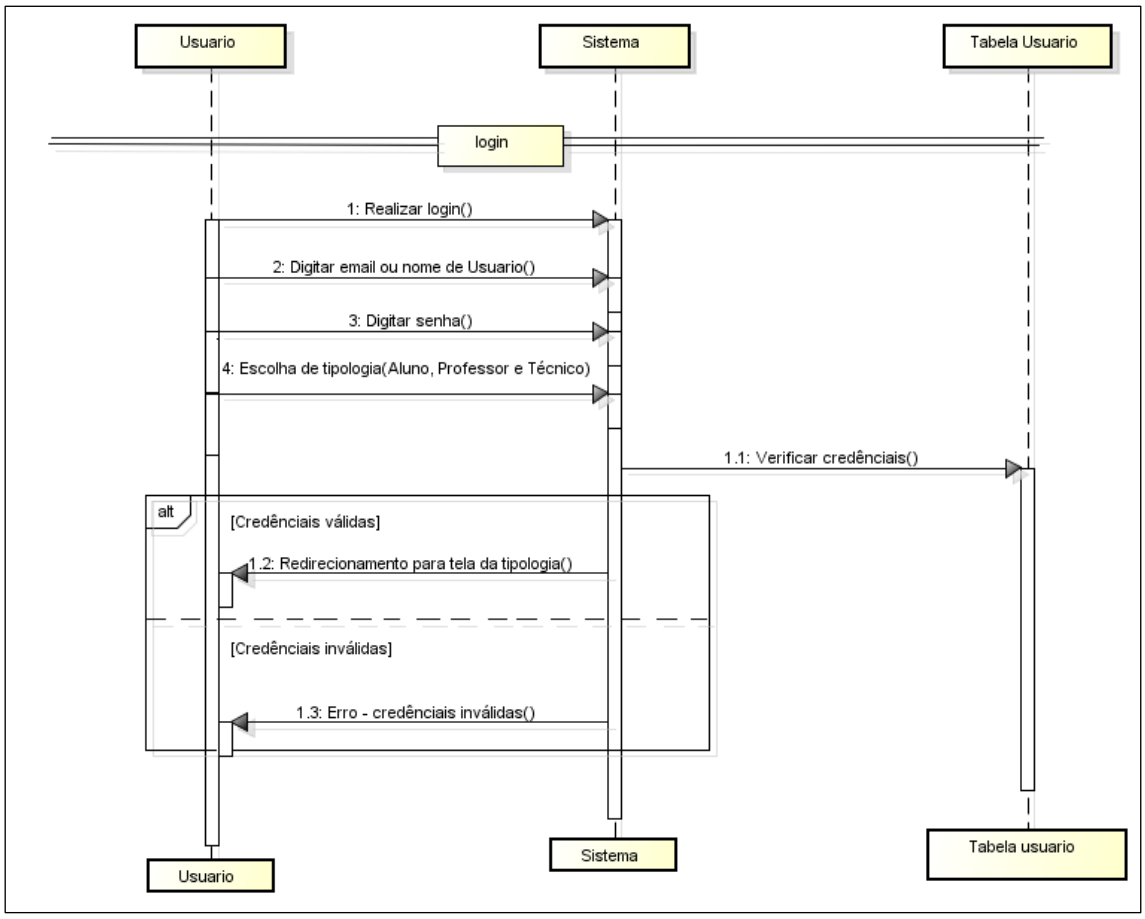
Fonte: Autoria Própria, 2023.

3.5 Diagrama de Sequência

O uso do diagrama de sequências como uma poderosa ferramenta visual para identificar alterações na ordem dos eventos, bem como compreender as relações entre objetos, classes e usuários em um determinado processo. O diagrama de sequências oferece uma representação clara das interações entre os elementos envolvidos, permitindo visualizar como as mensagens que são trocadas e como a informação é transmitida e processada.

O processo de login é uma etapa fundamental, este diagrama ilustra as interações entre os objetos envolvidos no processo, mostrando a sequência de mensagens trocadas entre eles. Permitindo que usuários acessem seus perfis. A figura 54, fornece um guia para o login do usuário através de um diagrama de sequência utilizando a notação da *Unified Modeling Language* (UML). Através de um estudo de caso prático, demonstraremos como identificar os objetos envolvidos, as mensagens trocadas e a sequência de interações como realizar o login demasiado a suas credenciais e-mail/nome de usuário e senha), o sistema enviar e verificar as credenciais para o banco e valida os dados, caso não seja válida ocorrerá um erro não permitindo o acesso, finalizando o sistema envia ao usuário a sua página inicial com base nos dados.

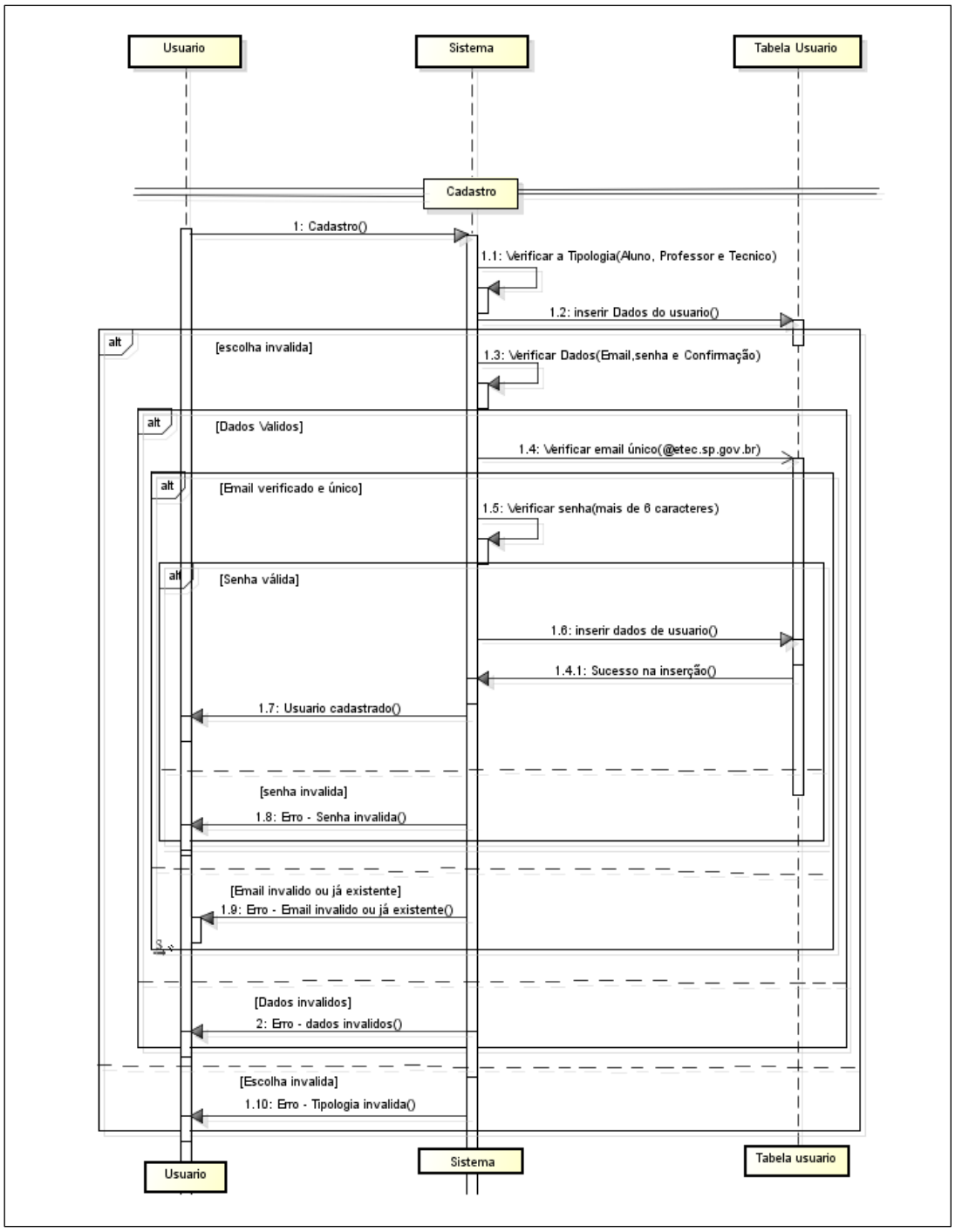
Figura 54 - Diagrama de Sequência do Login



Fonte: Autoria Própria, 2023.

A figura 55 representa um processo permitindo que novos usuários se registrem e obtenham acesso aos recursos disponíveis. Compreender o fluxo dessa funcionalidade é essencial para desenvolvedores e analistas de sistemas a fim de garantir a integridade dos dados e a eficiência do sistema como um todo, veja abaixo.

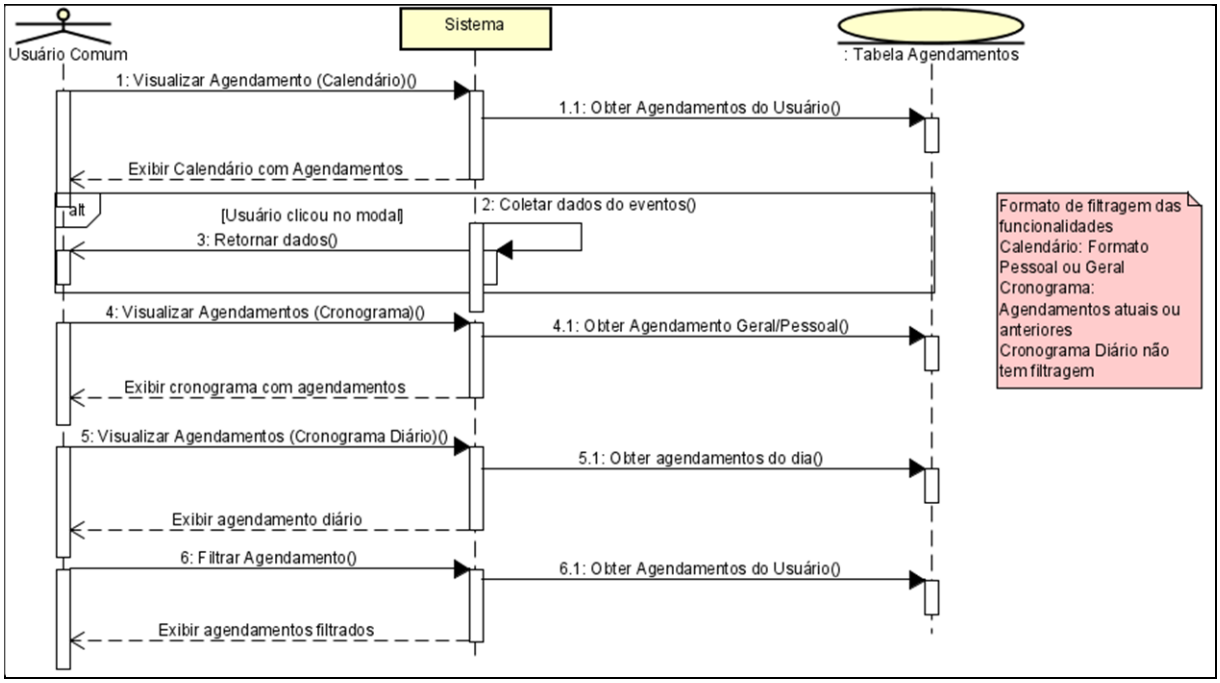
Figura 55 - Diagrama de Sequência do Cadastro



Fonte: Autoria Própria, 2023.

A figura 56, demonstra as possibilidades de visualização de um usuário ao receber os agendamentos, como por exemplo pelo Calendário, Cronograma e Agendamento do Dia (Cronograma diário). Todo exceto o último descrito tem sua forma de filtragem comum/geral

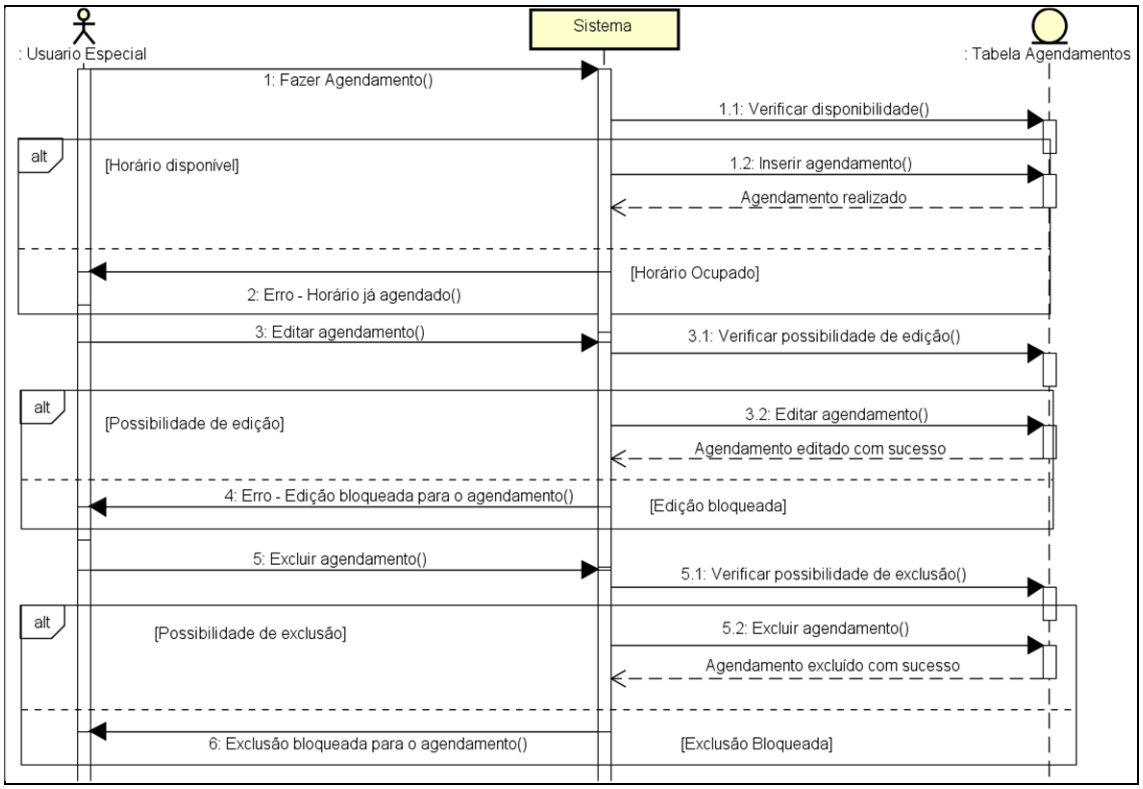
Figura 56 - Diagrama de Sequência de Receber Agendamentos



Fonte: Autoria Própria, 2023.

Pela figura 57, é possível ver a relação de uma forma mais ampla do Usuário Especial (autor exclusivo da funcionalidade no Caso de Uso) e sua relação com a tabela Agendamentos, sendo o Sistema a ponte para a relação dos 2, como para a verificação da disponibilidade, inserção ou alteração do agendamento e também a verificação "extra”

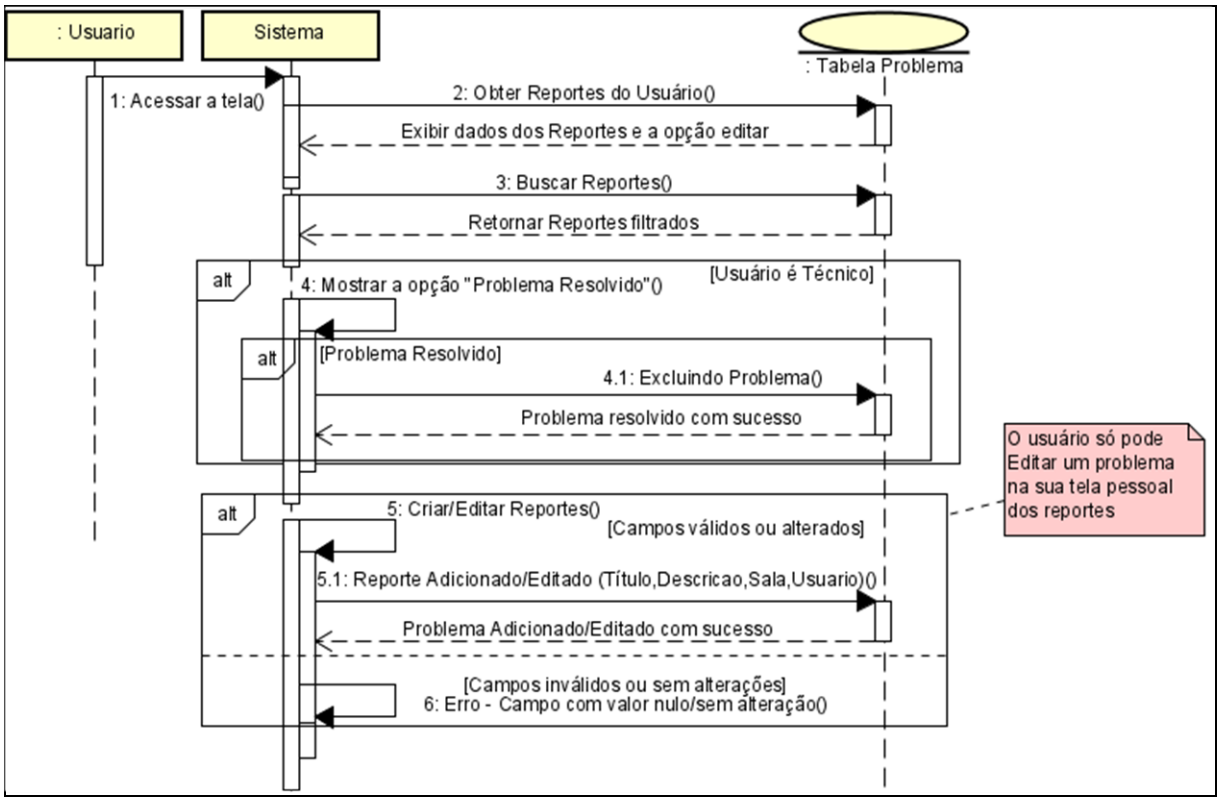
Figura 57 - Diagrama de Sequência de Manipular Agendamento



Fonte: Autoria Própria, 2023.

A figura 58 segue a mesma lógica da figura anterior, mas desta vez a relação é entre o sistema e os problemas, e entre o usuário e o sistema. Dessa forma, ele possui as mesmas funcionalidades descritas anteriormente, mas com a adição da possibilidade de filtrar os problemas pela visibilidade e pela sala e visualizá-los.

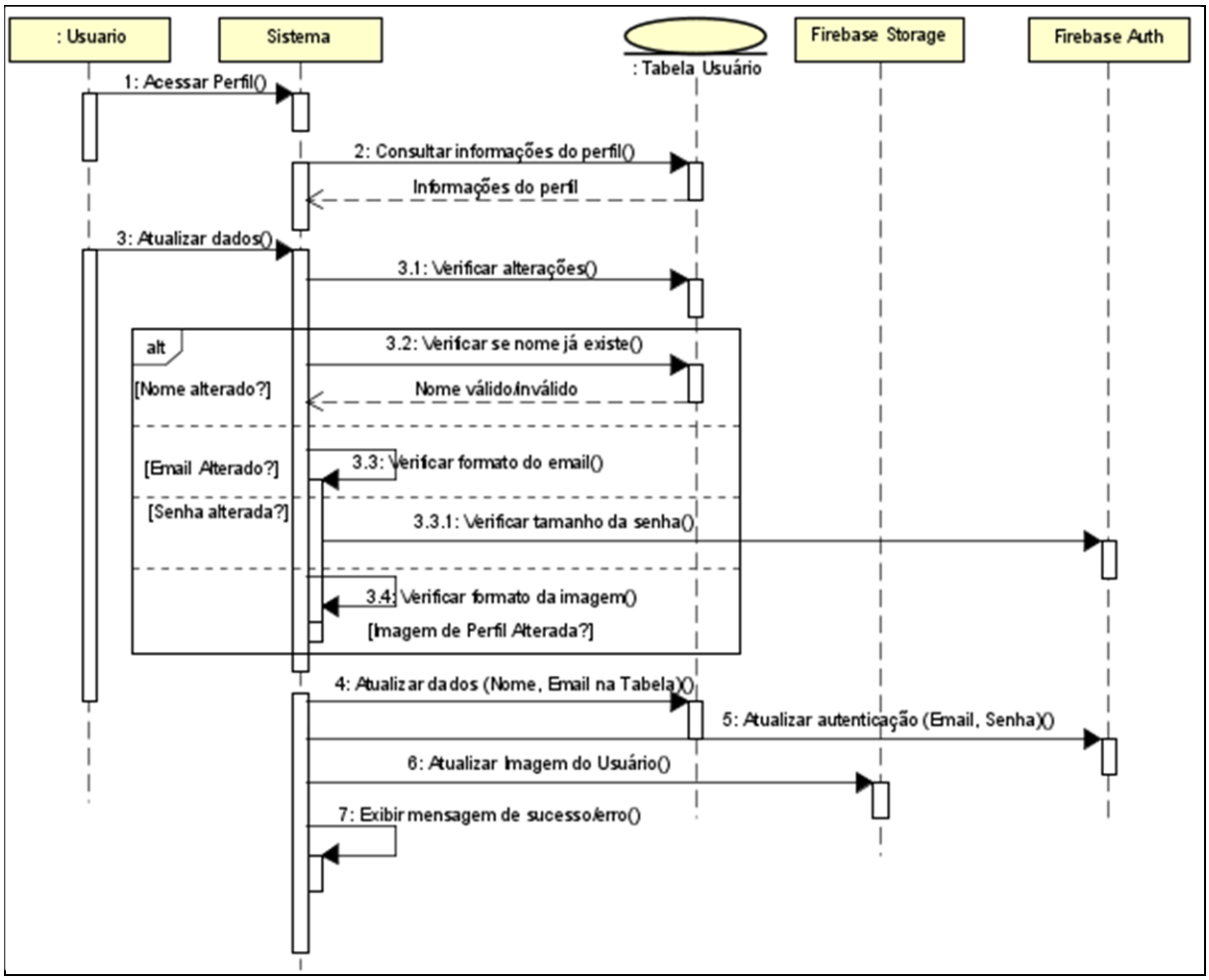
Figura 58 - Diagrama de Sequência do Reportar Problemas



Fonte: Autoria Própria, 2023.

A Figura 59 apresenta o diagrama de sequência que descreve as etapas do processo de formatação de perfil de usuário. O processo tem início com a solicitação de acesso ao perfil pelo usuário. Em seguida, o sistema verifica as permissões e autêntica o usuário. Isso permite que o usuário realize as formatações necessárias, de acordo com suas preferências individuais. Todas as informações modificadas são então armazenadas no banco de dados.

Figura 59 - Diagrama de Sequência do Reformular perfil



Fonte: Autoria Própria, 2023.

3.6 Aplicação

O aplicativo é composto de 25 telas, sendo que todas têm o seu modo padrão claro ou escuro e 17 referentes ao *dashboard* (área restrita ou de controle) para cada tipo de usuário, as quais se interagem pelas ações deles.

A primeira visível (figura 60) é a tela *home* do aplicativo em si, que têm um breve embasamento e resumo sobre o projeto, como os objetivos, opinião dos professores e uma imagem do mapeamento da unidade.

Figura 60 - Tela inicial do Site

Fonte: Autoria Própria, 2023.

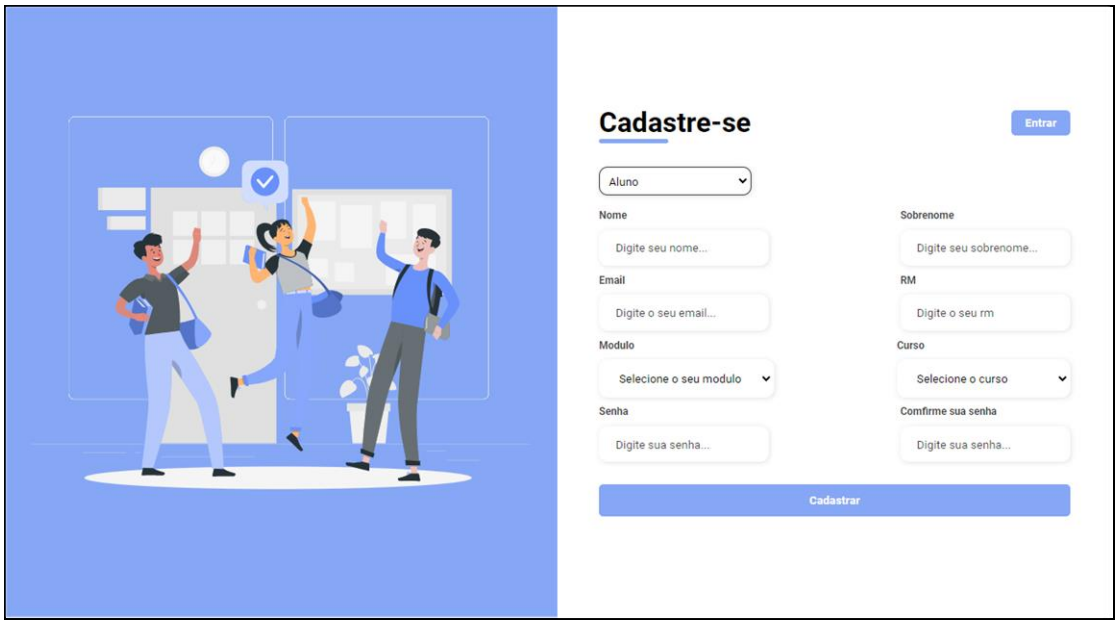
Seguindo em frente, temos 3 seções que se entrelaçam e muito: O Cadastro, *Login* e Esqueci a senha.

Como os próprios nomes já dizem, eles são a base para o usuário acessar sua área específica, seja aluno, professor ou técnico.

Para o usuário conseguir acessá-la, ele pode ir pelo *Login* (figura 65), mas caso ainda não tenha uma conta, ele necessita se cadastrar primeiro para poder fazer isso (figura 60). No cadastro ambos as tipologias (Aluno, Professor e Técnico) têm a opção do tipo do usuário em específico, em que no caso do Cadastro, existe um formulário avulso para cada um dos 3, como demonstrado nas figuras 62, 63 e 64.

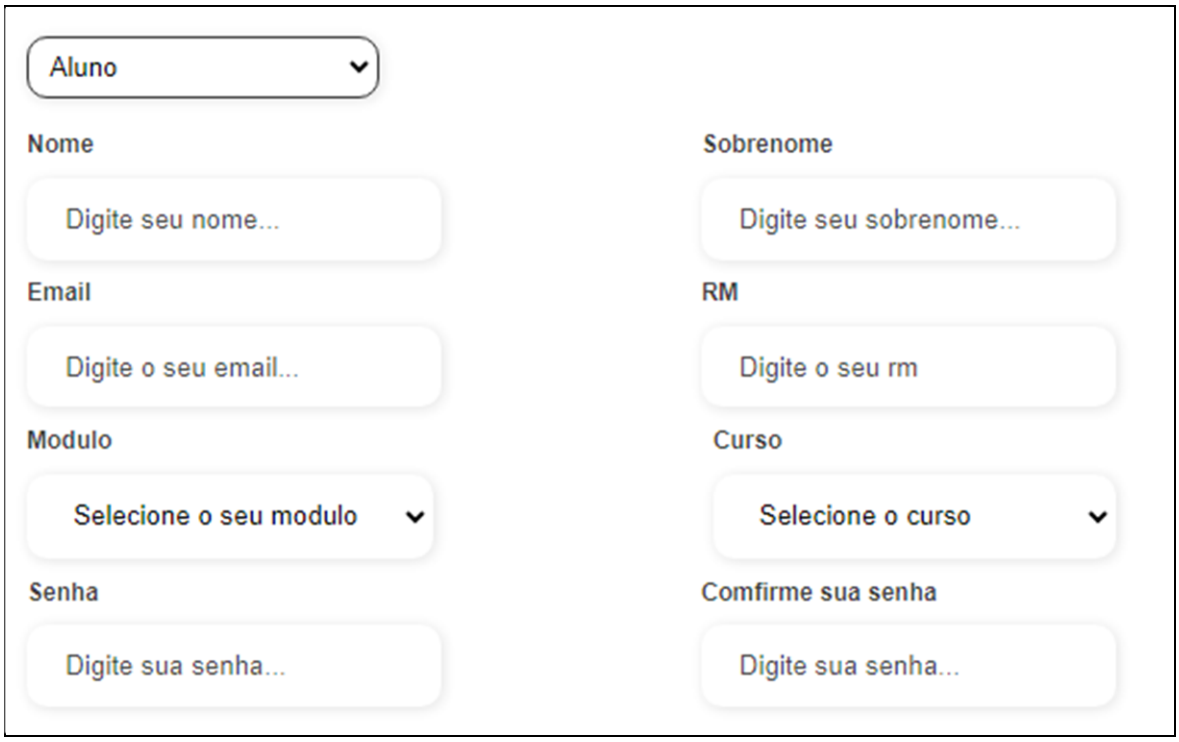
Referente ao esqueci a senha (figura 66), ela é uma tela avulsa para caso o usuário tenha criado a conta e não se lembre se sua senha. Caso ocorra isso, o usuário pode colocar o e-mail utilizado atualmente na conta, para assim ser mandado um e-mail para a alteração de senha no endereço indicado.

Figura 61 - Tela de Cadastro



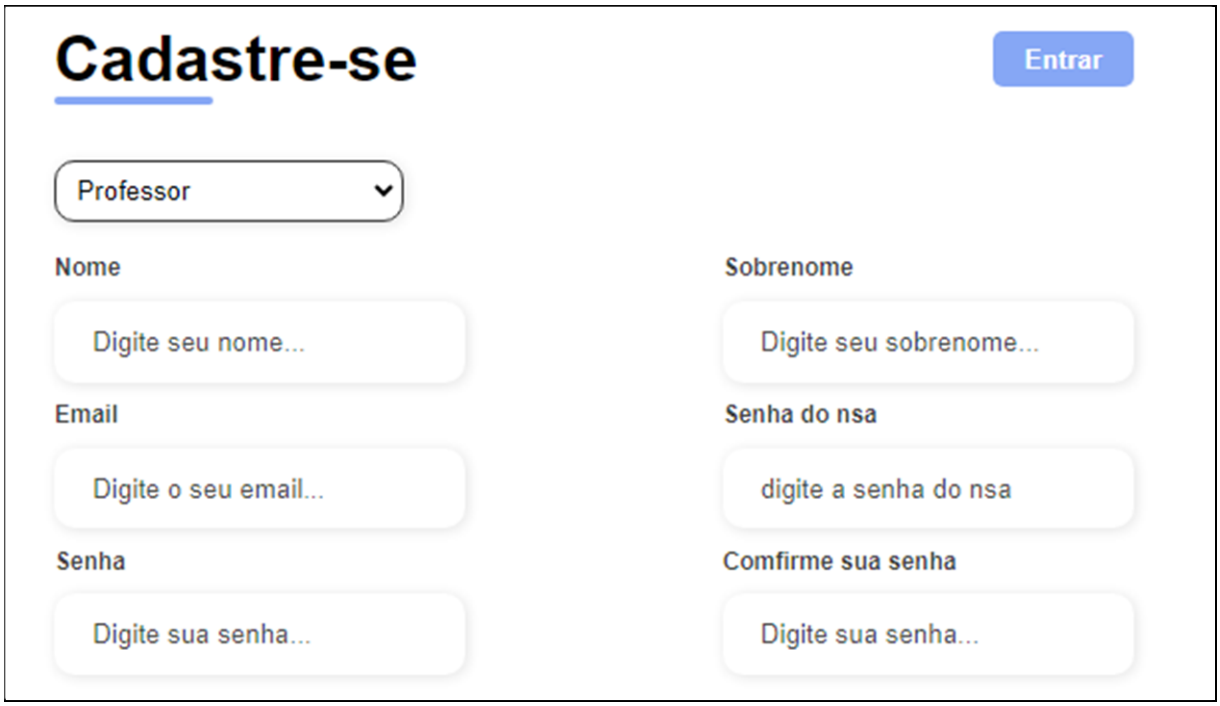
Fonte: Autoria Própria, 2023.

Figura 62 - Cadastro Aluno



Fonte: Autoria Própria, 2023.

Figura 63 - Cadastro Professor

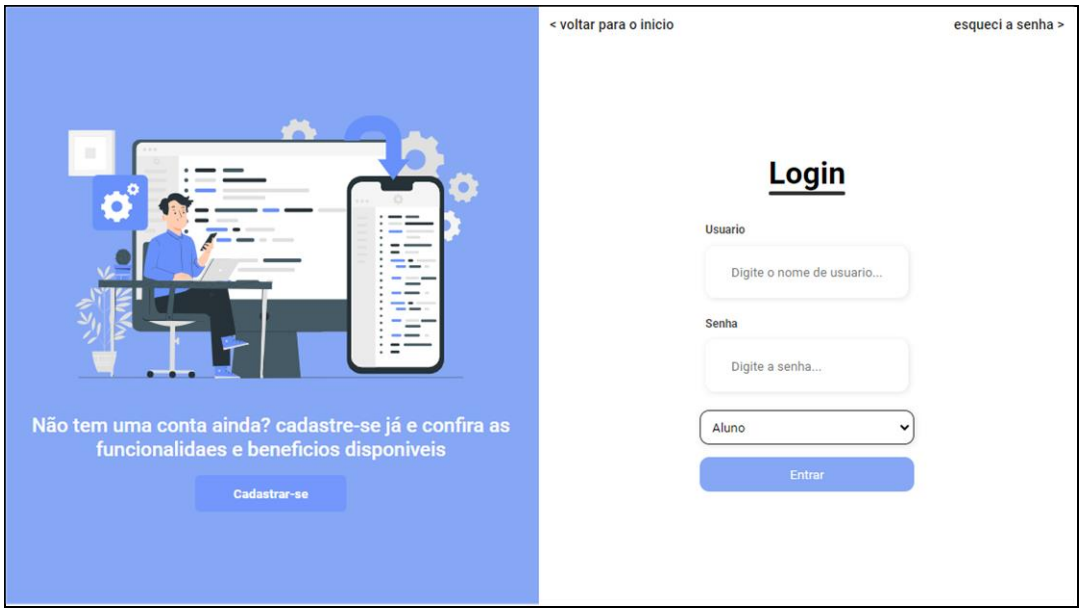


Fonte: Autoria Própria, 2023.

Figura 64 - Cadastro Técnico

Fonte: Autoria Própria, 2023.

Figura 65 - Login



Fonte: Autoria Própria, 2023.

Figura 66 - Esqueci a Senha

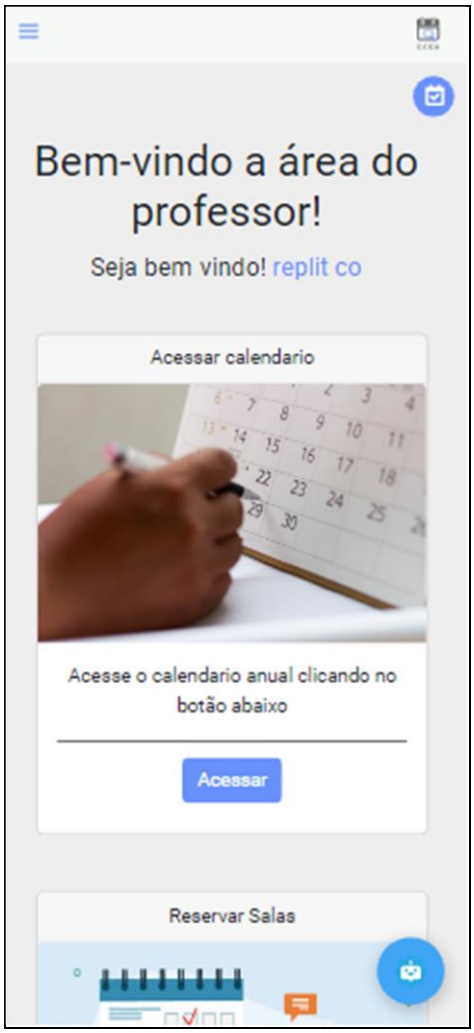
Fonte: Autoria Própria, 2023.

Adiante, começamos a seguir pelo *dashboard* do usuário. Na figura 67 temos uma visão de como é a tela do professor, no entanto, as outras telas, como a de aluno e técnico, contêm mudanças bem sutis, para combinar com as funcionalidades e utilidades de cada um, como no caso dos *cards* e da mensagem recebida.

No canto inferior direito contém uma “bolinha” que serve para o *chatbot* (figura 68), que será explicado melhor mais para frente, enquanto na superior (figura 69), existe a visualização do cronograma de atual do dia para o usuário no ícone de calendário.

A partir de agora, nas próximas telas, com exceção do perfil, estará listado no extremo canto esquerdo uma barra lateral (também chamado de *sidebar*) em que caso o usuário clique no ícone no extremo superior direito, será expandido e mostrará todas as funcionalidades que o usuário pode utilizar (figura 70), como os exemplificados anteriormente no Caso de Uso.

Figura 67 - Tela Inicial do Usuário

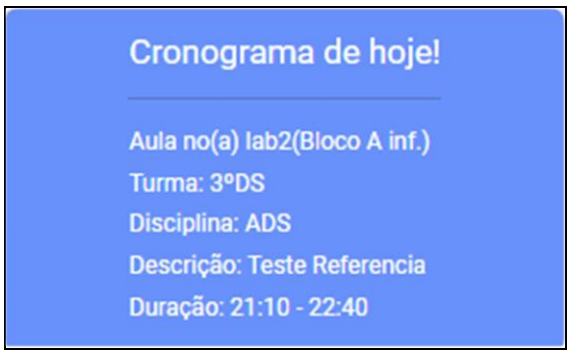


Fonte: Autoria Própria, 2023.

Figura 68 - Chatbot

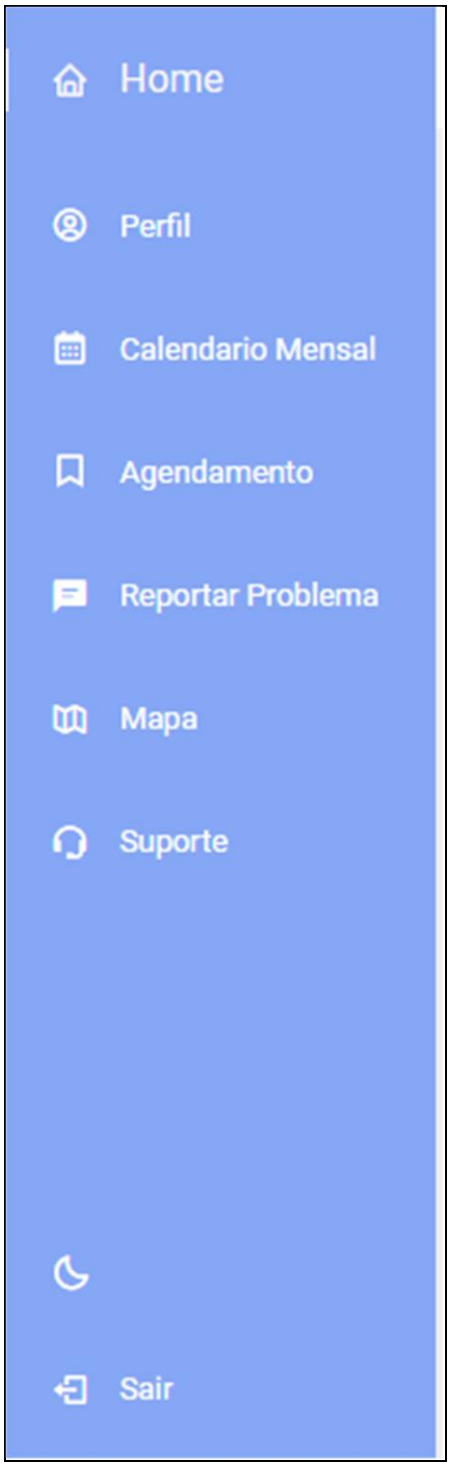
Fonte: Autoria Própria, 2023.

Figura 69 - Cronograma diário



Fonte: Autoria Própria, 2023.

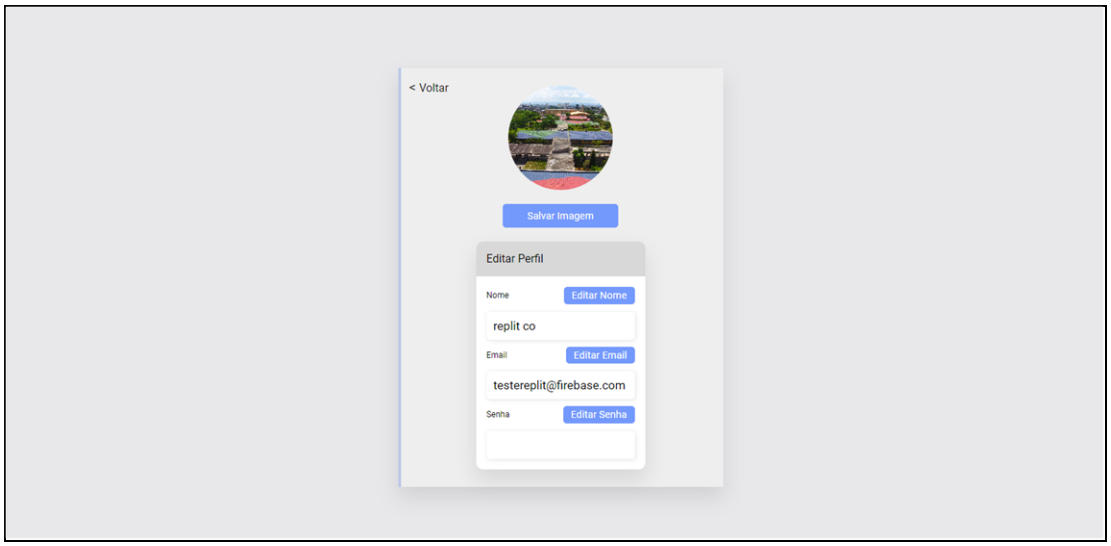
Figura 70 - Sidebar do dashboard



Fonte: Autoria Própria, 2023.

Na figura abaixo temos o Perfil, que contém a área de personalização do usuário, em que ele pode mudar a sua imagem de perfil, seu nome, sobrenome, e-mail, ou até mesmo a senha

Figura 71 - Perfil

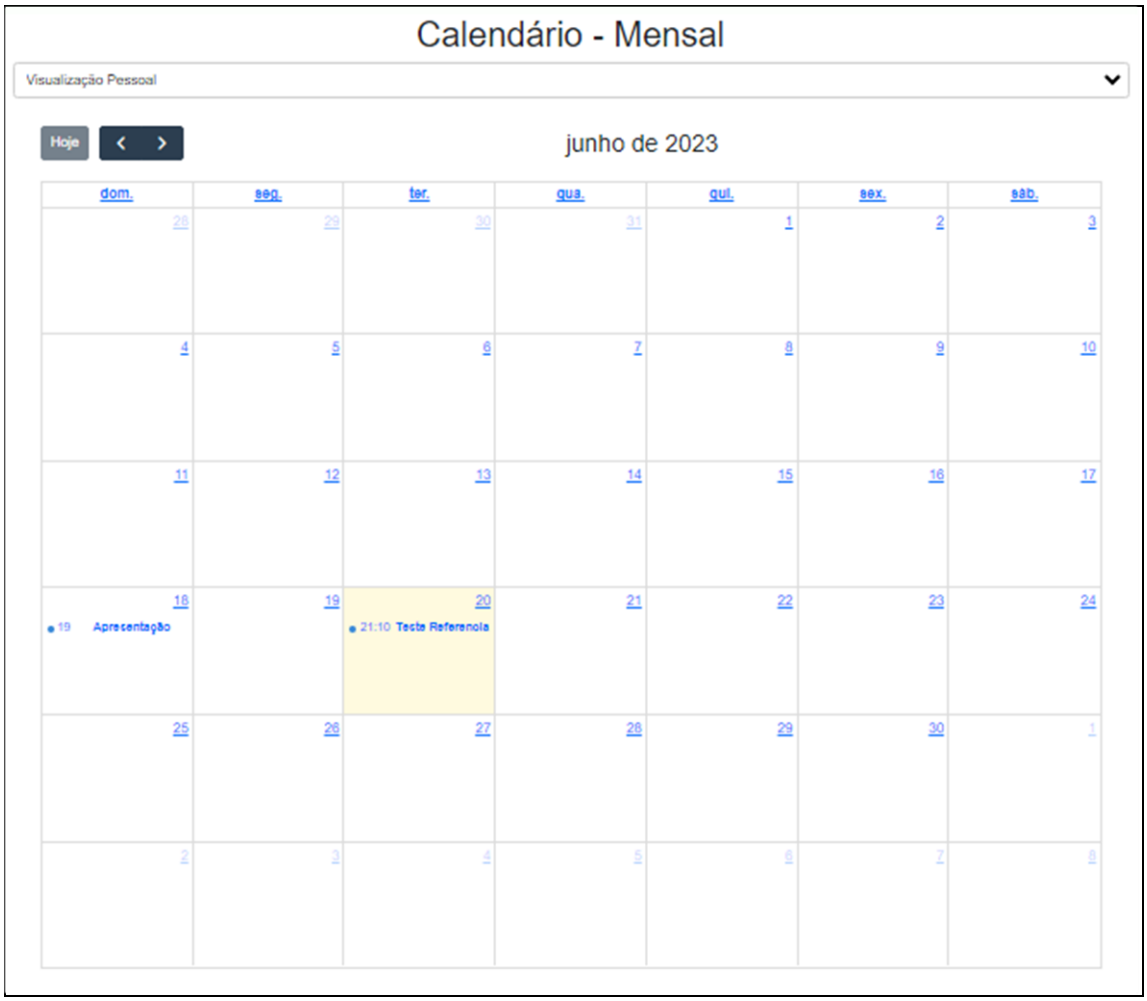


Fonte: Autoria Própria, 2023.

No calendário (figura 72), temos uma visualização bem mais clara e concisa do cronograma, em que caso o usuário clique em um evento específico, será impresso um *modal* (uma espécie de janela por cima da anterior, como no exemplo da figura 73) que especifica todos os dados de agendamento dos professores ou técnicos, seguindo a filtragem de cada tipo de usuário em específico

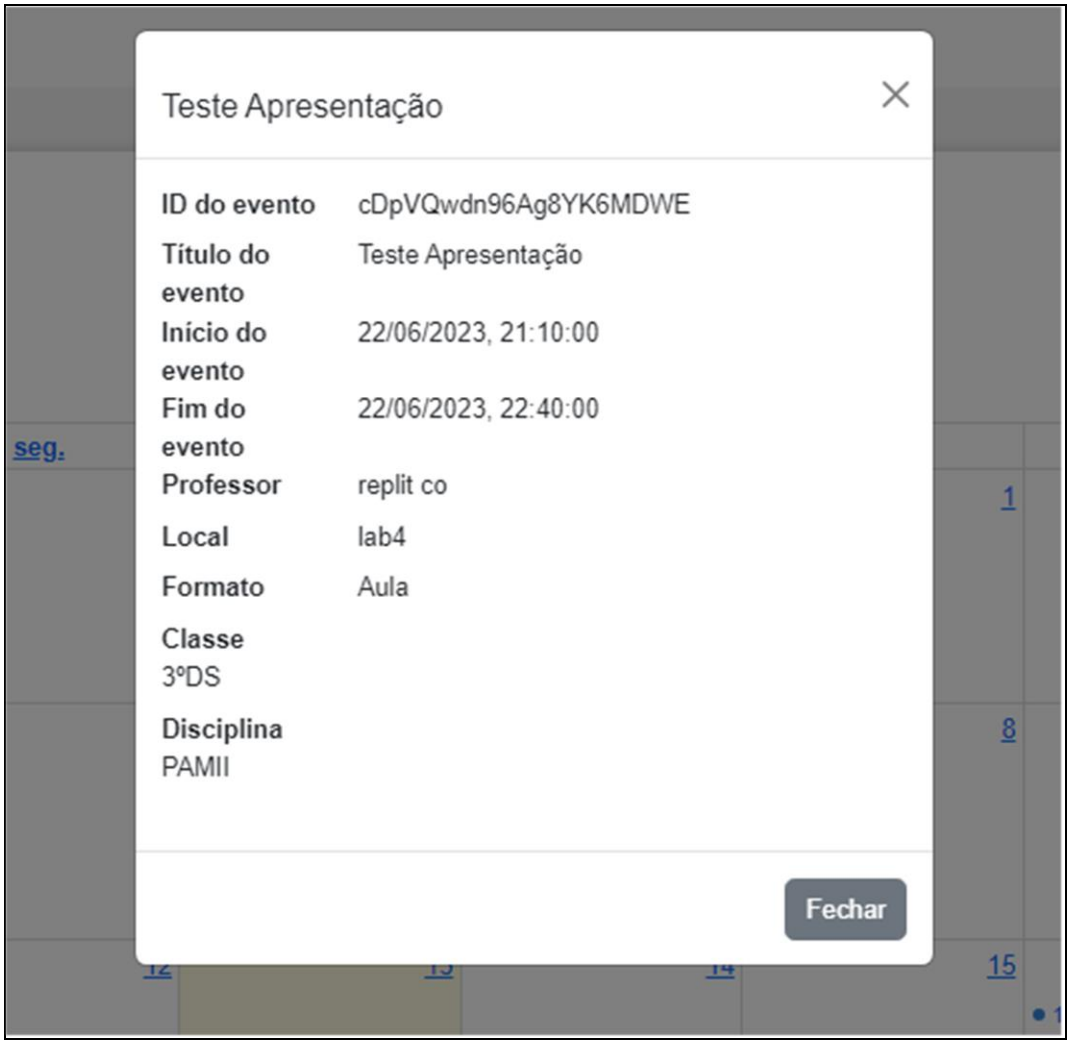
Na filtragem, o calendário contém a visualização pessoal (que depende da tipologia do Usuário) e a visualização geral, que conta todos os agendamentos feitos ao decorrer e que não foram removidos pelo seu autor.

Figura 72 - Calendário



Fonte: Autoria Própria, 2023.

Figura 73 - Modal do Evento



Fonte: Autoria Própria, 2023.

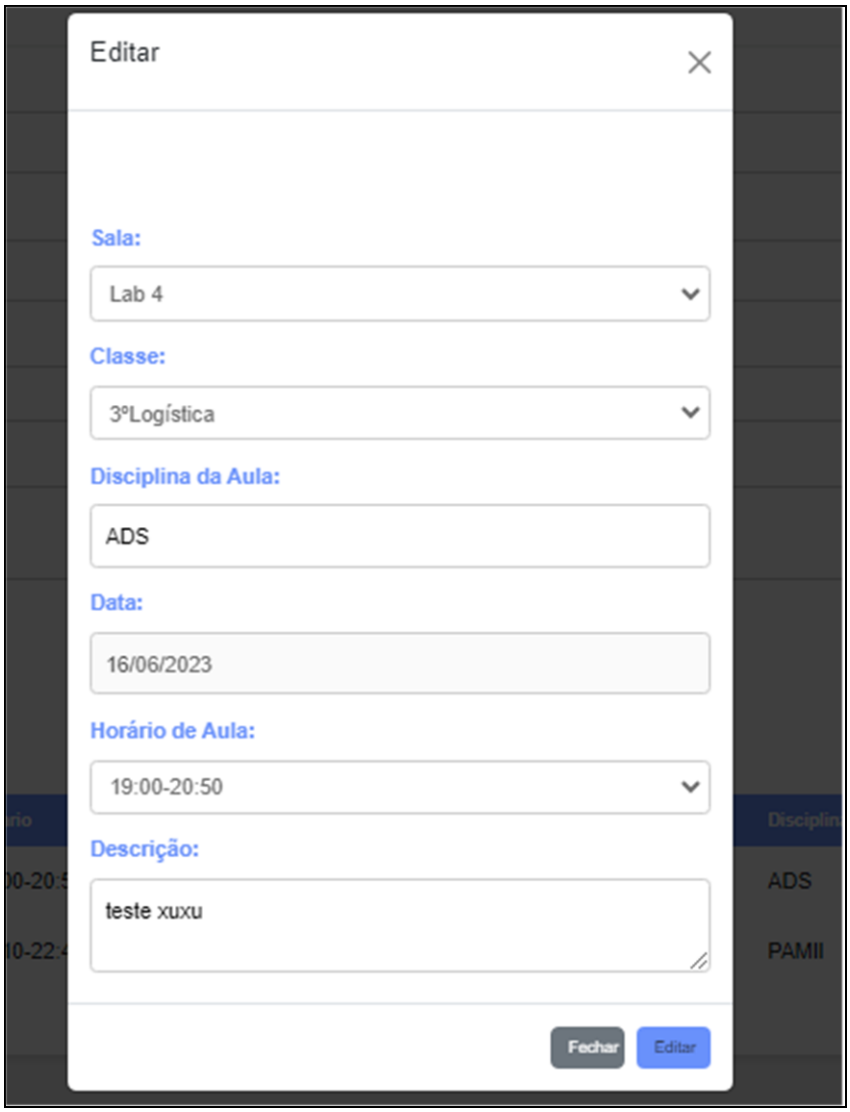
Na tela agendamento (figura 74), é a tela exclusiva do Técnico e Professor, ou seja, os Alunos não podem acessar essa parte. Nessa seção, o usuário faz o seu agendamento, o qual inclui a Sala, Data, horário para os 2, classe e disciplina especificamente para o Professor. Assim, se for um sucesso ou caso já exista agendamentos feitos pelo mesmo, aparecerá uma listagem desses, em que é possível editar (figura 75) ou excluir (figura 76) o específico.

Na filtragem do Agendamento e do Crongrama (tabela que fica abaixo da tela), o formato de visualização é pelos próximos agendamentos e anteriores feitos e ainda salvos (que é verificado pelo dia que o usuário acessou as telas), sendo que a diferença ficará mais clara quando for explicado essa próxima tela (Cronograma).

Figura 74 - Agendamento

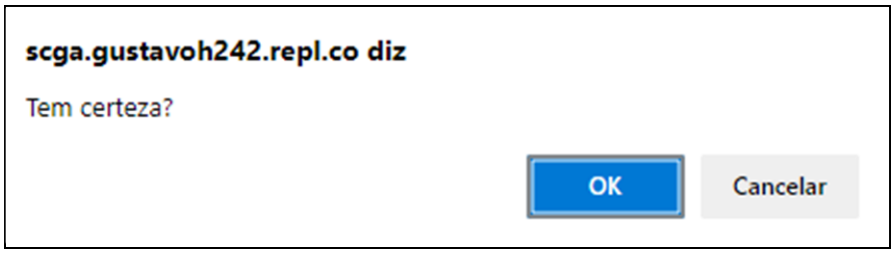
Fonte: Autoria Própria, 2023.

Figura 75 - Edição do Agendamento



Fonte: Autoria Própria, 2023.

Figura 76 - Exclusão do Agendamento

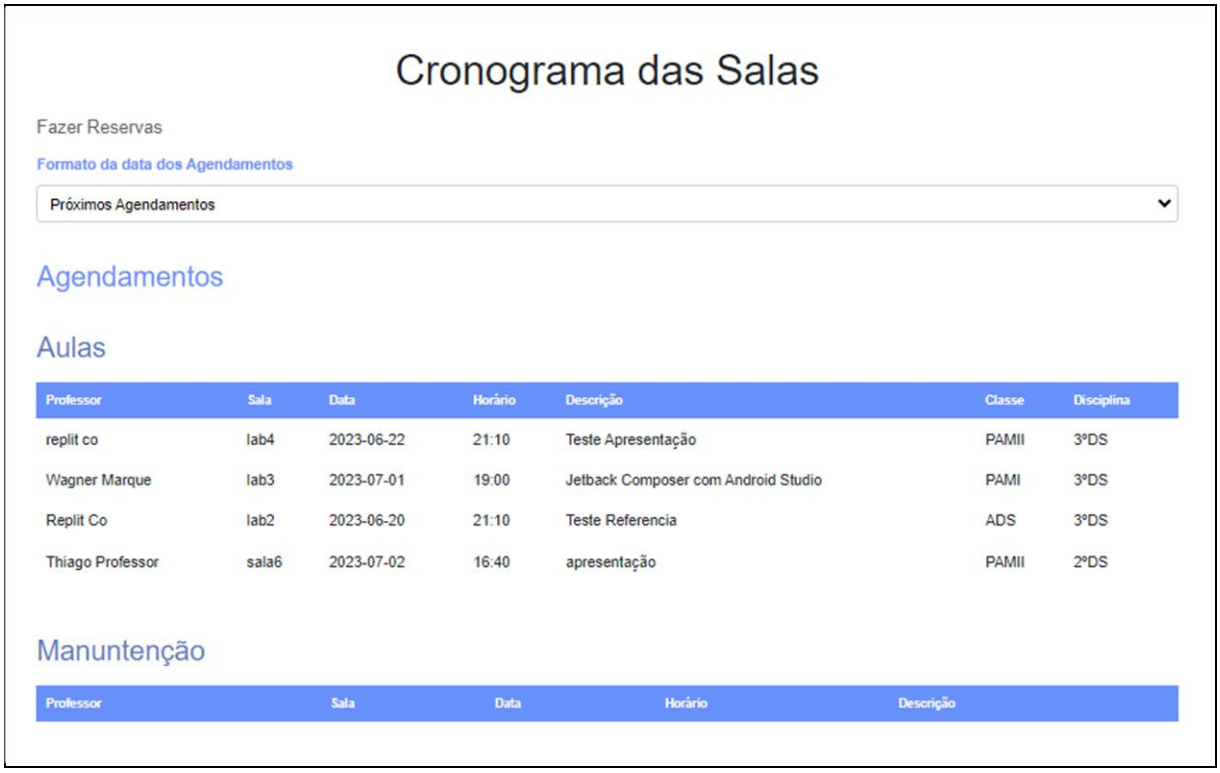


Fonte: Autoria Própria, 2023.

Referente ao Cronograma (figura 77), ele tem uma visualização distinta do calendário e do agendamento, dividindo cada agendamento em uma linha através de uma tabela como no caso da funcionalidade anterior, englobando mais que os agendamentos do

usuário em específico. Por ser uma visualização mais geral, a tabela é dividida em 2: uma para os Agendamentos dos Professores (Aulas) e outra para dos técnicos (Manuntenção).

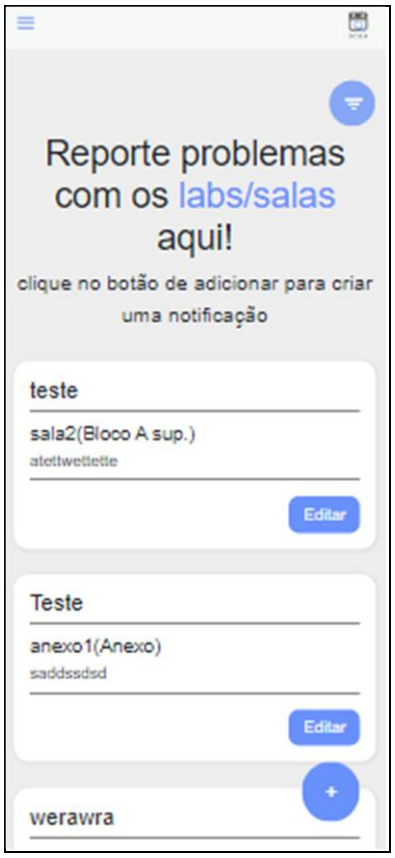
Figura 77 - Cronograma



Fonte: Autoria Própria, 2023.

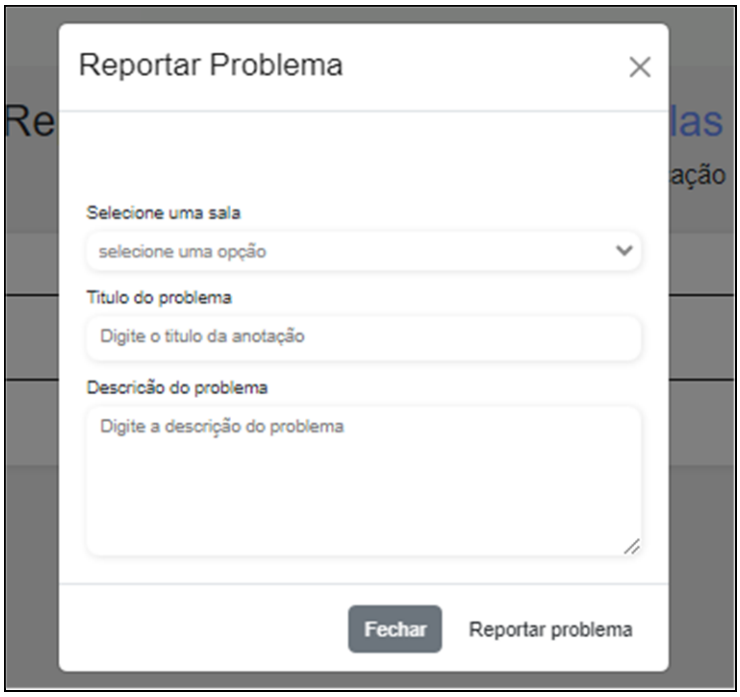
O Reportar problemas serve para o usuário poder fazer suas reclamações (figura 78), referente à problemas de equipamentos ou componentes de um ambiente em específico. Ambos os tipos de usuários podem fazer sua reportagem de problemas (figura 79), edição (figura 80) e filtragem (figura 81), respectivamente pelo ícone “+”, botão “editar” e de filtro pelo canto superior direito, em que, no entanto, somente o técnico poderá limpar ou contabilizar se o problema foi resolvido.

Figura 78 - Reportar Problemas



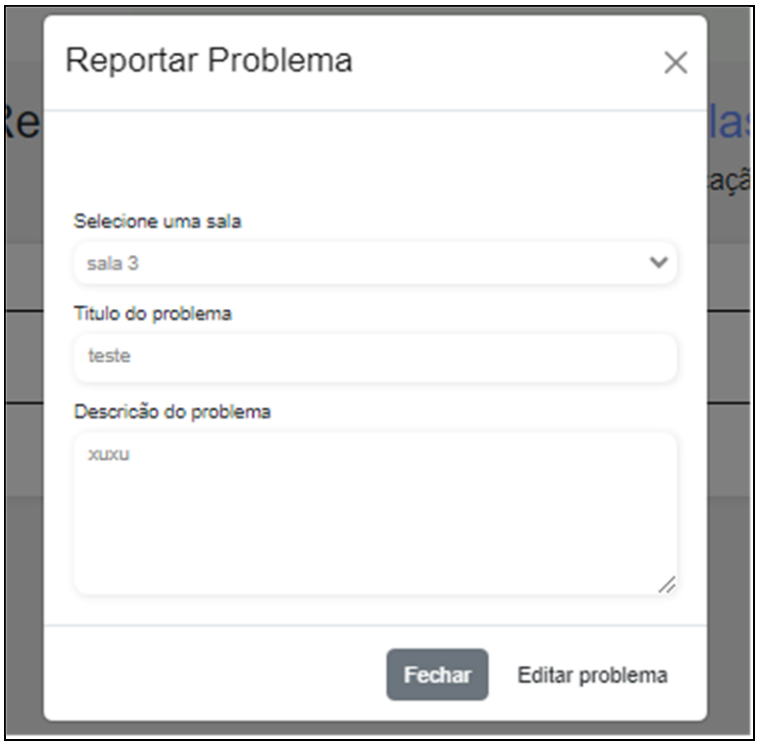
Fonte: Autoria Própria, 2023.

Figura 79 - Criação do Problema



Fonte: Autoria Própria, 2023.

Figura 80 - Edição do Problema



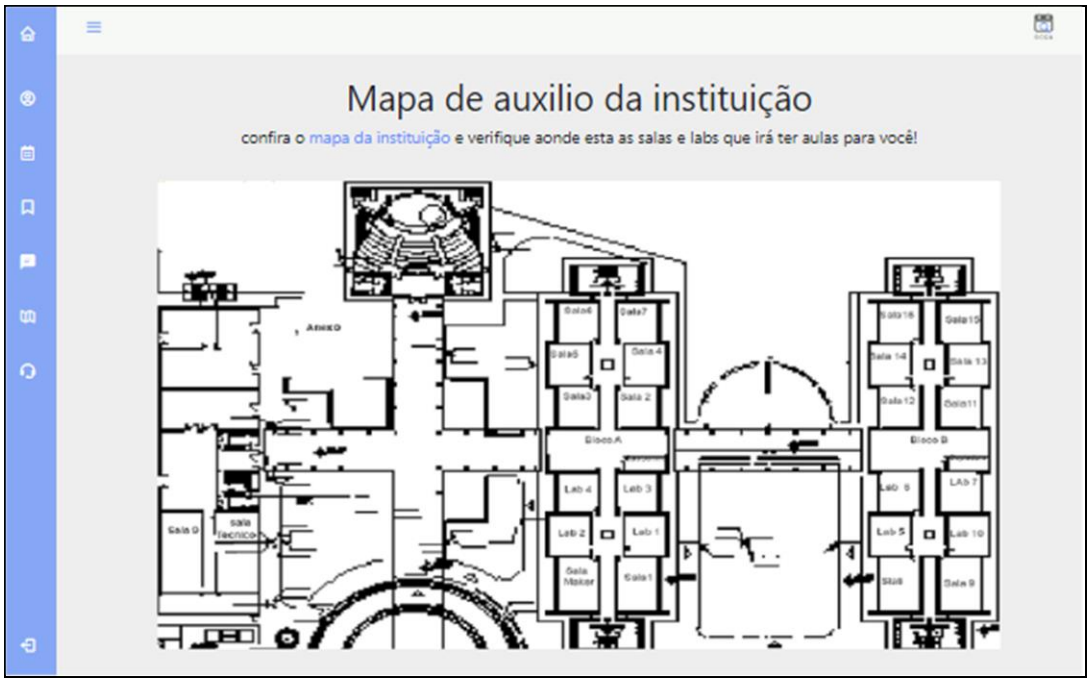
Fonte: Autoria Própria, 2023.

Figura 81 - Filtragem dos Problemas

Fonte: Autoria Própria, 2023.

Para uma visão mais detalhada e específica para o Usuário, na área restrita também temos uma tela que demonstra o mapeamento da Unidade, mostrando as localizações dos Blocos, salas, labs e derivados como a figura 82 demonstra.

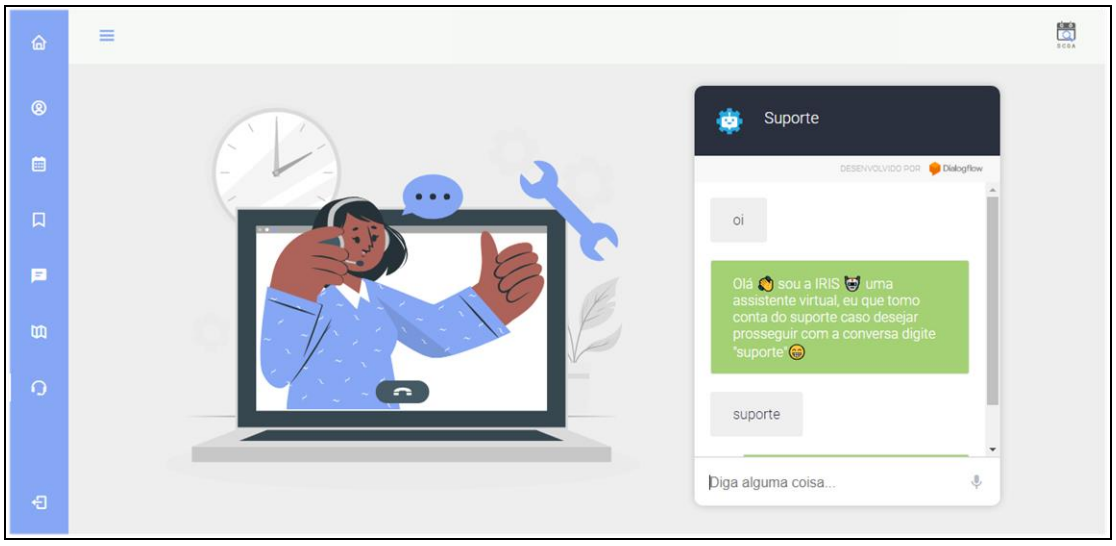
Figura 82 - Mapa



Fonte: Autoria Própria, 2023.

No suporte (figura 83), temos o *chatbot* (*software* que simula a interação humana em canais e plataformas) que foi brevemente citado na tela inicial do usuário. Ele contém diversas funcionalidades, como explicar a história do site e da Etec, assim como da localização de determinadas funcionalidades caso o usuário se confunda no momento.

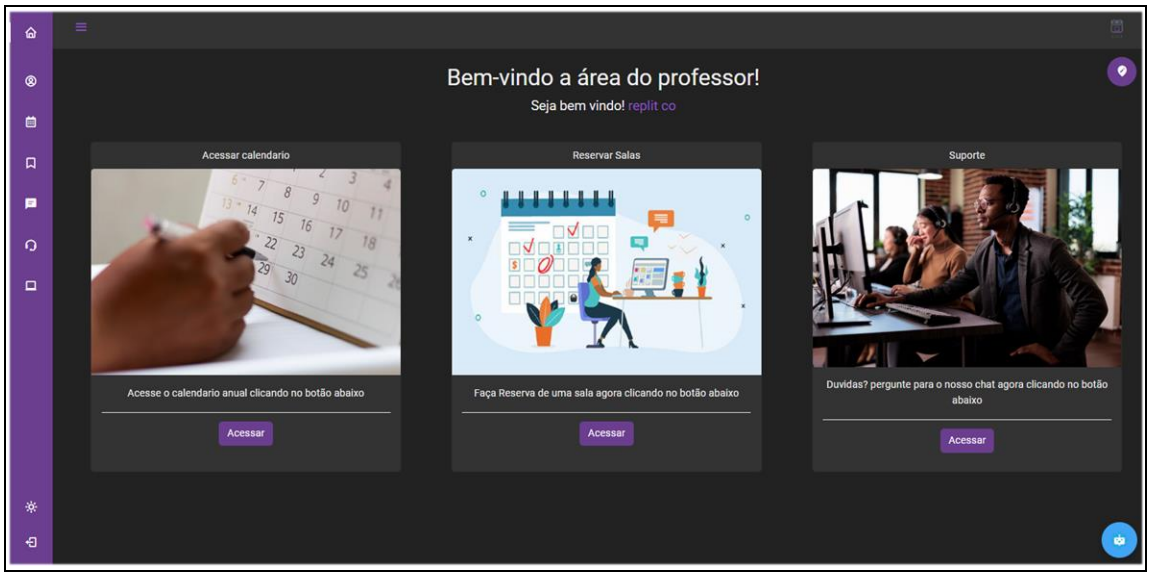
Figura 83 - Suporte



Fonte: Autoria Própria, 2023.

Finalizando, temos também a funcionalidade de alteração do tema do site na tela inicial do site e do usuário, pelo ícone do sol ou da lua. Abaixo, temos uma demonstração breve da diferença do tema padrão (branco) para o escuro (dark) na figura 84, em que os itens que eram azuis se tornam roxos, o branco fica escuro e virse-versa.

Figura 84 - Tema Escuro



Fonte: Autoria Própria, 2023.

4 CONSIDERAÇÕES FINAIS

O desenvolvimento deste projeto nos proporcionou uma valiosa experiência de aprendizado e a oportunidade de enfrentar desafios reais relacionados à comunicação. Ao longo desse processo, aplicamos os conhecimentos adquiridos tanto internamente quanto externamente ao curso, buscando solucionar problemas que afetam o âmbito educacional de gerenciamento de ambientes entre usuários que acessam e utilizam a unidade.

Nossa plataforma se mostrou uma ferramenta funcional e flexível, capaz de facilitar a comunicação e auxiliar na resolução dessas questões. Além disso, percebemos que esse projeto nos permitiu amadurecer não apenas no campo da tecnologia da informação, mas também no entendimento da importância da comunicação e organização em diversos contextos, além ter sido gratificante criar uma plataforma que realmente pudesse ser funcional e capaz de auxiliar e beneficiar muitas pessoas de diferentes campos, mas de um só local, como o nosso estimado professor Wagner havia salientado anteriormente.

Com base nessa experiência, temos a firme intenção de continuar desenvolvendo projetos futuros e melhorando cada vez mais esse, explorando diferentes linguagens e sistemas, para criar soluções que atendam a uma ampla variedade de necessidades de comunicação. Nosso objetivo principal é tornar os problemas e desafios enfrentados por diferentes grupos mais acessíveis, proporcionando uma mitigação das dificuldades e contribuindo para um ambiente mais harmonioso e eficiente.

REFERÊNCIAS

ÁLVAREZ, Miguel Angel et al. **Manual de CSS 3**. 2017

BIZELLI, José Luís; DA CUNHA DARIDO, Maíra. **Inovações tecnológicas e contexto escolar**: reflexões necessárias. Revista Ibero-Americana de Estudos em Educação, v. 10, n. 1, p. 50-66, 2015

BOOCH, Grady. **UML: guia do usuário**. Elsevier Brasil, 2006.

CARVALHO, Vítor Manuel Ferreira Alves. **HTML**: curso de introdução. 1997.

CRISPINIANO, Almir Gonçalves. **ESTUDO COMPARATIVO ENTRE FRAMEWORKS FRONTEND PARA A CRIAÇÃO DE UM PROGRESSIVE WEB APP (PWA)**. 2021.

CLOUD, Google. **Documentação do Firebase**. 2021. Disponível em: <https:// firebase.google.com/docs/>

CONVERSE, Tim; PARK, Joyce. **PHP: a bíblia**. Gulf Professional Publishing, 2003.

CURVELLO, João José Azevedo. Estudos de comunicação organizacional: entre a análise e a prescrição. In: **conferencia], XXV Congresso Brasileiro de Ciéncias da Comu-Nicaçâo-Intercom-NP Relaçôes Públicas e Comunicaçâo Organizacional [CD-ROM], Salvador/BA**. 2002.

DALL'OGLIO, Pablo. **PHP Programando com Orientação a Objetos 3ª Edição**. Novatec Editora, 2015.

DA SILVA, Wildemarkes de Almeida et al. **Google Forms como ferramenta para avaliação da aprendizagem**. 2018.

DE ALMEIDA, Vinicius Coelho. **Uso da linguagem OCL no contexto de diagramas de classe da UML e programas em Java**. 2006.

DE LEMOS, Maxmilian Ferreira et al. **Aplicabilidade da arquitetura MVC em uma aplicação web (WebApps)**. RE3C-Revista Eletrônica Científica de Ciência da Computação, v. 8, n. 1, 2013.

DOUGLAS, Michael; MARABESI, Matheus. Aprendendo Laravel: O framework PHP dos artesãos da web. Novatec Editora, 2017.

ESPINDOLA, Maisa Baêta; OLIVEIRA, Annévia Palhares Vieira Diniz. **Análise comportamental**: um estudo de como o comportamento organizacional pode influenciar o clima organizacional de uma indústria. SYNTHESIS| *Revistal Digital FAPAM*, v. 1, n. 1, p. 179-200, 2009.

FLATSCHART, Fábio. **HTML 5-Embarque Imediato**. Brasport, 2011.

Fernando Henrique Canto, Augusto Dias Pereira dos Santos, **Workshop de Tecnologia da Informação das IFES** (3.: 2009: Belém). [Anais..]. Belém: UFPA, 2009.

FOWLER, Martin. **UML Essencial: um breve guia para linguagem padrão**. Bookman editora, 2014.

GABARDO, Ademir C. **Laravel para ninjas**, Novatec Editora, 2017.

Gaunt, M. Web Fundamentals: **Introdução aos Service Workers**. 2021 de Agosto. Disponível em: <https://developers.google.com/web/fundamentals/primers/service-workers/>

GUEDES, Gilleanes TA. **UML 2-Uma abordagem prática**. Novatec Editora,2018.

JÚNIOR, Edwar Saliba. **Diagrama de Caso de Uso**. 2020.

JÚNIOR, Edwar Saliba. **Diagrama de Classes**. 2020.

LAGOA, Alda Maria Monteiro Costa. **A comunicação interna como ferramenta de gestão** - um Estudo de caso no agrupamento de escolas Fontes Pereira de Melo. 2021. Tese de Doutorado.

LEAL, Regina Barros. Planejamento de ensino: peculiaridades significativas. **Revista Iberoamericana de Educación**, v. 37, n. 3, p. 1-6, 2005.

LEWIS, Joseph R.; MOSCOVITZ, Meitar. **CSS avançado**. Tradução de Edgard B, p. 16, 2010.

LIMA, Matheus. **Como criar seu primeiro Progressive Web App do Zero**. 2017. Disponível em <https://medium.com/tableless/como-criar-seu-primeiro-progressive-web-app-do-zero-a7e6ca5fb21e>. Acesso em: 27 Mar 2017

LÓSCIO, Bernadette Farias; OLIVEIRA, HR de; PONTES, JC de S. **NoSQL no desenvolvimento de aplicações Web colaborativas**. VIII Simpósio Brasileiro de Sistemas Colaborativos, v. 10, n. 1, p. 11, 2011.

MACHADO, Fhabiana Thieli dos Santos et al. **Um processo para extração de esquemas conceituais em fontes de dados JSON baseado em técnicas de similaridade de texto**. 2017.

MALHOTRA, Naresh K. **Pesquisa de marketing: uma orientação aplicada**. Bookman Editora, 2001.

MARCHIORI, Marlene. **Cultura e comunicação organizacional: um olhar estratégico sobre a organização**. Difusão Editora, 2018.

MAZZAROLO, Vinícius; DA SILVA, Roger Sá. **Progressive Web Apps**: Uma nova abordagem no desenvolvimento de aplicações Web, 2021

MURTA, Leonardo Gresta Paulino. **Diagrama de Atividades**. 2015

ODORICO, Elizandra K. et al**. Análise do não uso do laboratório de informática nas escolas públicas e estudo de caso**. In: Anais do XVIII Workshop de Informática na Escola. SBC, 2012. p. 38-46.

PAULO, Vandré. **O que é CSS**. 2007. Disponível em: <https://web.archive.org/web/20130117151242/http://revistaphp.com.br/artigo.php?id=119/>. Acesso em: 21 maio 2007.

PELIZZA, Angelica Caetane; BERTOLINI, Cristiano; SILVEIRA, Sidnei Renato. **Um estudo sobre técnicas de teste de software no Framework Laravel**. Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica, v. 9, n. 3, 2018.

PEREIRA, Peter Paul; REBOLO, Flavines. Clima escolar e suas implicações para o trabalho docente. Série-Estudos, v. 22, n. 46, p. 93-112, 2017.

PRESCOTT, Preston (ed.). **Programação em JavaScript**. Online: Babelclube, 2016.

PINHEIRO, Paulo César da Costa. **De um tutorial hipertexto em HTML**. In: XXV Congresso Brasileiro de Ensino de Engenharia (COBENGE-97). 1997. p. 12-15.

PIOVESAN, Armando; TEMPORINI, Edméa Rita. Pesquisa exploratória: procedimento metodológico para o estudo de fatores humanos no campo da saúde pública. **Revista de saúde pública**, v. 29, p. 318-325, 1995.

SANTIAGO, Cynthia Pinheiro et al. **Desenvolvimento de sistemas Web orientado a reuso com Python, Django e Bootstrap**. Sociedade Brasileira de Computação, 2020.

SCHWARTZMAN, Simon. O Centro Paula Souza e a educação profissional no Brasil. **Educação Básica no Estado de São Paulo**: avanços e desafios. São Paulo: Fundação Seade, p. 187-216, 2014.

SILVA, Ângela Carrancho da. **Educação e tecnologia: entre o discurso e a prática. Ensaio: avaliação e políticas públicas em educação**, v. 19, n. 72, p. 527-554, 2011.

SILVA, Paulo Caetano da; SILVA JÚNIOR, João Batista da. **Análise da representação semântica de modelos de dados do formato JSON**. Revista de Sistemas e Computação-RSC, v. 8, n. 1, 2018.

SMITH, Ben. **JSON básico: conheça o formato de dados preferido da web**. Novatec Editora, 2020.

SOUZA, Ivan. **JavaScript**: o que é, como funciona e por que usá-lo no seu site. 2019. Disponível em: https://rockcontent.com/br/blog/javascript. Acesso em: 23 nov. 2020.

SPUDEIT, Daniela. **Elaboração do plano de ensino e do plano de aula**. Rio de janeiro, 2014.

TORRES, Victor Monteiro. **HTML e seus Componentes**. Revista Ada Lovelace, v. 2, p. 99-101, 2018.

TRINDADE, Patrícia Esteves; AFFINI, Letícia Passos. **APONTAMENTO A CERCA DO PROGRESSIVE WEB APPS**, 2018.

VINHA, Telma Pileggi et al. **O clima escolar e a convivência respeitosa nas instituições educativas**. Estudos em Avaliação Educacional, v. 27, n. 64, p. 96-127, 2016.

Gudwin, Ricardo R. "Introdução à linguagem UML." www. dca. fee. unicamp. br/~ gudwin/ftp/ea976/Estruturais2010. pdf>. Acesso em 15.08 (2010): 2014

.

Júnior, Edwar Saliba. "**Diagrama de Classes.**" (2020).

Bezerra, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. Vol. 2. Rio de Janeiro: Elsevier, 2007.