CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ETEC DA ZONA LESTE

Desenvolvimentos de Sistemas

Felipe Eduardo Garcia Nunes de Souza

João Victor Cahuaya Mayta

Marinho José Fernandes Junior

Matheus Lopes Oliveira

Nicolas dos Santos Ramos

DEVSCHOOL: Metodologia de ensino à programação

São Paulo

2023

Felipe Eduardo Garcia Nunes de Souza

João Victor Cahuaya Mayta

Marinho José Fernandes Junior

Matheus Lopes Oliveira

Nicolas dos Santos Ramos

DEVSCHOOL: Metodologia de ensino à programação

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Desenvolvimento de Sistemas da Etec Zona Leste, orientado pelo Prof. Ediney Ciasi Barreto, como requisito parcial para obtenção do título de técnico em Desenvolvimento de Sistemas.

São Paulo

2023

DEDICATÓRIA

Dedicamos este trabalho aos nossos pais e familiares, aos colegas, e aos professores.

AGRADECIMENTOS

Agradeço aos meus professores e colegas por me ajudarem a desenvolver este trabalho

EPÍGRAFE

“Não amanhã, não semana que vem, nem no mês que vem. Você precisa tomar uma decisão hoje de que está pronto para fazer as mudanças necessárias para garantir que será capaz de criar a vida que de fato deseja.”

HAL ELROD

RESUMO

O presente trabalho tem como tema a didática, tanto da programação, quanto minimizar um dos maiores vilões no processo de aprendizagem dos jovens, o desvio e desinteresse no estudo diário. O objetivo é compreender as dificuldades dos jovens no aprendizado da programação, entendendo os problemas dessa área, fazendo com que ele evolua e se sinta motivado nos estudos de uma forma que seja interativo, buscando melhorar a eficácia e constância na aprendizagem e estudo da área, deixando de delongar o aprendizado de modo que use métodos úteis para um bom desempenho nos estudos. O uso da programação tem sido cada vez mais requisitado, principalmente por jovens que querem entrar no mercado de trabalho. A introdução de aulas interativas tem o potencial de transformar o ambiente educacional e melhorar o ensino-aprendizagem, além de proporcionar uma experiência mais agradável para os alunos na sala de aula. Essa abordagem também incentiva os educadores a criarem um ambiente mais envolvente e interessante, resultando em um maior prazer em ensinar. Assim, considera-se todo o estudo da linguagem iniciativa com recursos didáticos viáveis e práticos para quem quer aprender programação, permitindo uma boa escalação de aprendizagem do usuário.

Palavras-chave: Programação; Ensino; Aprendizado; Incentivo.

ABSTRACT

The present study focuses on the didactics of programming and minimizing one of the biggest obstacles to the learning process of young people, which is deviation and disinterest in daily studies. The objective is to understand the difficulties that young people face in learning programming, identifying the problems in this field and helping them evolve and feel motivated in their studies in an interactive way, seeking to improve the effectiveness and consistency of learning and studying in this area, and avoiding prolonging the learning process by using useful methods for good performance in studies. The use of programming has become increasingly demanded, especially by young people who want to enter the job market. The introduction of interactive classes has the potential to transform the educational environment and improve the teaching-learning process, as well as provide a more enjoyable experience for students in the classroom. This approach also encourages educators to create a more engaging and interesting environment, resulting in greater satisfaction in teaching. Thus, the study considers the language initiative with viable and practical didactic resources for those who want to learn programming, allowing for a good escalation of the user's learning.

Keywords: *Programming; Teaching; Learning; Motivation.*

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

Administrador de banco de dados (DBA)

Application Programming Interface (API)

Banco de Dados (BD)

Cascading Style Sheets (CSS)

Centro Universitário de Belo Horizonte (UniBH)

Diagrama de Entidade e Relacionamento (DER)

European Computer Manufacturer's Association (ECMA)

Extensible Markup Language (XML)

JavaScript (JS)

Hypertext Markup Language (HTML)

Hypertext Preprocessor (PHP)

Inteligência Artificial (IA)

Modelo de Entidade e Relacionamento (MER)

Modelo de Objeto de Documento (DOM)

Not Only SQL (NoSQL)

Pixels (PX)

Red Green Blue (RGB).

Sistemas gerenciadores de banco de dados (SGBD)

Structured Query Language (SQL)

Unified Modeling Language (UML)

Uniform Resource Locator (URL)

SUMÁRIO

1 INTRODUÇÃO

É fato que o mercado de trabalho na programação é o que mais cresce nos últimos anos, tendo em mente que a demanda só aumenta, a certos problemas que para entender os conceitos da área dificulta o progresso de novatos, não sabendo por onde ou como começar, principalmente quando chegam em um curso seja técnico ou superior.

A utilização da programação e de outras tecnologias, aliada à aplicação de técnicas de ludificação, possui o potencial de impulsionar significativamente o engajamento dos alunos no processo de aprendizagem. Em contraste ao ensino tradicional, que muitas vezes é considerado entediante e pouco estimulante para determinados indivíduos, um site de curso intuitivo, baseado em IA como o Chat GPT (*generative pre-trained transformer*, ou transformador pre-treinado generativo, em português), pode oferecer uma experiência de aprendizagem mais atrativa e envolvente.

Segundo Borges (2000), a área de programação apresenta um alto índice de alunos que finalizam o curso sem um conhecimento mínimo adequado. Além disso, muitos alunos abandonam o curso já nos primeiros semestres devido às dificuldades encontradas.

Prior (MOREIRA, 2009), diz que a capacidade de programar não pode ser desenvolvida sem a realização de atividades práticas em laboratório. Para ajudar no ensino de programação prática, várias ferramentas foram propostas para auxiliar os educadores. No entanto, mesmo com o surgimento desses sistemas de ensino de programação, alguns obstáculos ainda são enfrentados pelos professores, como a de avaliar todos os exercícios de todos os alunos rapidamente. Geralmente, os sistemas permitem que os alunos criem seus algoritmos, mas exige muito do professor para fornecer feedback imediato e disponibilizar correções para todos os alunos.

Diante desse cenário, uma solução seria o uso de plataformas de aprendizado online que utilizam técnicas de participação, oferecendo atividades práticas e desafios que estimulam o engajamento dos alunos e o aprendizado em programação de forma mais eficiente e atrativa. É importante que as instituições de ensino invistam em soluções inovadoras que atendam às necessidades dos alunos e

ajudem a superar as barreiras enfrentadas no aprendizado da programação. A utilização de plataformas de aprendizado online pode ajudar a estimular o engajamento dos alunos e ajudar a melhorar às formas de entender conceitos básicos da programação, preparando-os para o mercado de trabalho em constante evolução da área.

2 REFERENCIAL TEÓRICO

Neste texto, serão apresentadas todas as tecnologias que foram utilizadas no projeto de finalização de curso. O objetivo é fornecer uma visão geral das ferramentas e tecnologias envolvidas no desenvolvimento do projeto, com o intuito de oferecer uma melhor compreensão do trabalho realizado.

2.1 HTML

De acordo com Elcio e Diego (2011), *Hypertext Markup Language* (HTML) que traduzindo para o português significa Linguagem de Marcação de Hipertexto. Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio etc.) para a *web*.

HTML é baseado no conceito de Hipertexto, que é uma forma de organizar conteúdo de forma não linear. Hipertexto são conjuntos de elementos – ou nós – ligados por conexões. Estes elementos podem ser palavras, imagens, vídeos, áudio, documentos etc. Estes elementos conectados formam uma grande rede de informação. Eles não estão conectados linearmente como se fossem textos de um livro, onde um assunto é ligado ao outro seguidamente. A conexão feita em um hipertexto é algo imprevisto que permite a comunicação de dados, organizando conhecimentos e guardando informações relacionadas. (FERREIRA; EIS, 2011, p. 26).

O HTML é muito utilizado para desenvolvimento *web*, navegação na internet e documentação.

Conforme Elcio e Diego (2011) essas *tags* têm o seguinte funcionamento e importância dentro do código:

- *Doctype*: deve estar na primeira linha do código. Indica ao navegador qual especificação de código que deve ser utilizada. Não é uma *tag,* mas sim uma

instrução para o *browser*, o seu navegador, ter informações de qual versão de código a marcação foi escrita.

- HTML: o código HTML é composto por vários elementos em árvore, onde uns são filhos dos outros. Nesta *tag* pode ser implementado a linguagem principal do documento a partir do atributo *lang.* Por exemplo: <html lang=”pt-br”>.
- *Head*: é onde se encontra a parte inteligente da página. Nesta ficam os metadados, que são informações sobre o conteúdo publicado e a página em si.
- *Title: tag* onde é escrito o título da sua página, o qual aparece na guia do seu navegador.
- *Meta charset*: é uma *metatag* encarregada por atribuir qual tabela de caracteres que a página utilizará.
- *Link*: existem dois tipos no HTML, um em que leva o usuário para outro documento e um para fontes externas usadas no documento. Neste exemplo a *tag link* importa o *Cascading Style Sheets* (CSS) para a página, o que indica esta importação é o atributo rel=”stylesheet”.
- *Body*: esta última indica o corpo da sua página. Todo o conteúdo da sua página deverá ficar dentro desta *tag.*

Segue em baixo a estrutura do HTML:

Figura 1 – Estrutura de um HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <title>Título da Página</title>
</head>
<body>

</body>
</html>
```

Fonte: Autoria própria, 2023.

Se aprofundando um pouco mais, podemos criar formulários interativos para coletar informações dos usuários. Para isso, usamos diversas *tags* que desempenham papéis específicos. A seguir, explicarei cada uma delas detalhadamente:

- <*Form*>: é usada para criar um formulário no HTML. Ela envolve todos os elementos do formulário, como campos de entrada, botões e textos explicativos. O atributo <*action*> pode ser usado para especificar o *Uniform Resource Locator* (URL) ou o *script* que receberá os dados do formulário quando ele for enviado.
- <*Input*>: é utilizada para criar campos de entrada no formulário. Existem vários tipos de campos, como texto, número, e-mail, senha, *checkbox*, *radio* <*button*>, entre outros. O atributo <*type*> define o tipo de campo de entrada que será exibido.
- <*Div*>: é uma *tag* de divisão genérica que é frequentemente usada para agrupar elementos relacionados em uma seção do formulário ou em um *layout* específico. Ela pode ser estilizada usando CSS para alterar a aparência e o posicionamento dos elementos contidos.
- <*Label*>: é usada para criar um rótulo ou etiqueta para um elemento de formulário. Ela geralmente é associada a um campo <*input*> usando o atributo *for*, que deve ter o mesmo valor que o atributo *id* do campo correspondente. Isso ajuda a melhorar a acessibilidade e usabilidade do formulário, permitindo que os usuários cliquem no rótulo para focar no campo de entrada.
-
: é usada para inserir uma quebra de linha, permitindo que o texto ou os elementos subsequentes sejam exibidos em uma nova linha. Ela é frequentemente usada para separar visualmente diferentes elementos de formulário ou criar espaçamento entre linhas.
- <*Textarea*>: é utilizada para criar uma área de texto expansível, onde os usuários podem inserir texto com várias linhas. Ela é especialmente útil para receber mensagens, comentários ou qualquer outro tipo de texto mais longo.
- <*Button*>: é usada para criar um botão no formulário. Ela pode ser usada para enviar o formulário, redefinir campos ou executar ações específicas definidas pelo desenvolvedor.

- <h1>: o H1 possui um destaque maior, uma fonte maior, e é geralmente o elemento de texto mais visível da página.
- <p>: é usada para criar um parágrafo de texto. Representa "parágrafo", essa *tag* é usada para separar o conteúdo em blocos de texto e é uma das *tags* mais comumente usadas para formatação de texto em HTML.

Com essas *tags*, é possível criar um formulário simples e funcional em HTML, coletando informações dos usuários e permitindo a interação com o site. A combinação e o uso adequado dessas *tags* ajudam a criar uma experiência amigável e intuitiva para os usuários.

Veja um exemplo de como ficaria um formulário simples usando tudo que foi mostrado:

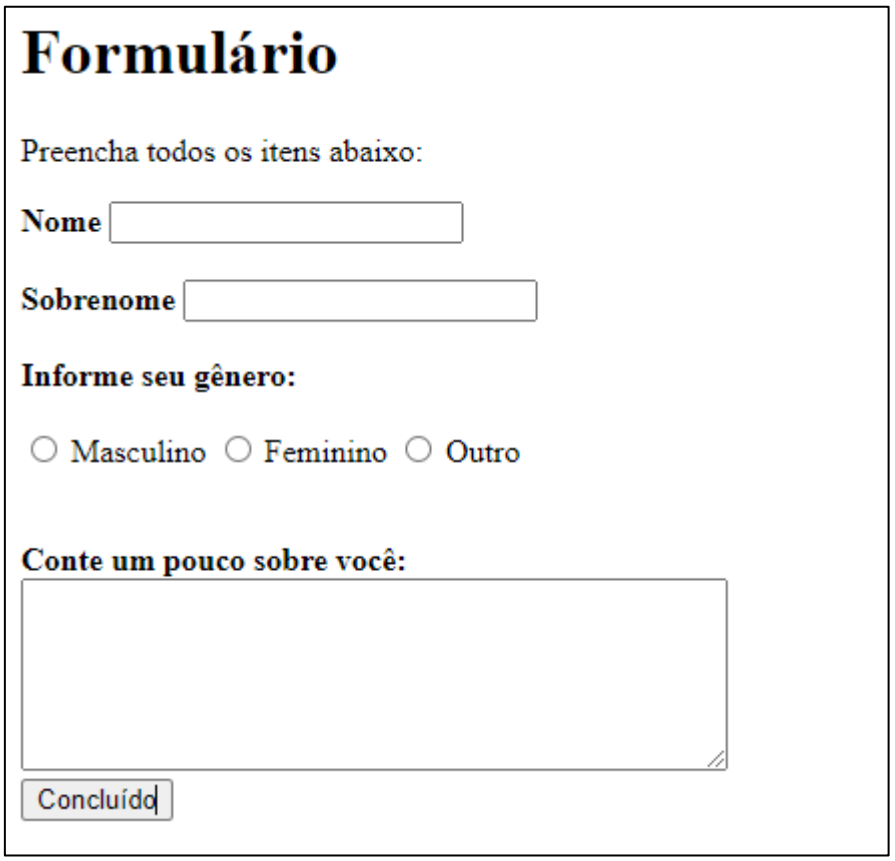
Figura 2 – Exemplo de Formulário em HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <title>Formulario</title>
</head>
<body>
    <h1> Formulário </h1>
    <p> Preencha todos os itens abaixo: </p>
    <form>
        <div>
            <label for="nome"><strong>Nome</strong></label>
            <input type="text" name="nome" id="nome" required>
        </div>
    </form></br>
    <div>
        <label for="sobrenome"><strong>Sobrenome</strong></label>
        <input type="text" name="sobrenome" id="sobrenome" required>
    </div><br>
    <div id="check">
        <label><strong>Informe seu gênero:</strong></label><br><br>
        <input type="radio" id="Masculino" name="Masculino" value="Masculino">
        <label for="Masculino"> Masculino</label>
        <input type="radio" id="Feminino" name="Feminino" value="Feminino">
        <label for="Feminino"> Feminino</label>
        <input type="radio" id="Outro" name="Outro" value="Outro">
        <label for="Outro"> Outro</label>
    </div><br>
    <div>
        <br>
        <label for="experiencia"><strong>Conte um pouco sobre você: </strong></label> </br>
        <textarea rows="6" style="width: 26em" id="experiencia" name="experiencia"></textarea>
    </div>
    <button class="botao" type="submit" onsubmit="">Concluído</button>
    </form>
</body>
</html>
```

Fonte: Autoria própria, 2023.

A seguir o resultado do código acima em uma página *web*:

Figura 3 – Exemplo Formulário



Fonte: Autoria própria, 2023.

2.2 CSS

O *Cascading Style Sheets* (CSS) formata a informação que é entregue pelo HTML. Essa informação pode ser qualquer coisa: imagem, texto, vídeo, áudio ou qualquer outro elemento criado. Grave isso: CSS formata a informação (FERREIRA; EIS, 2011, p. 137).

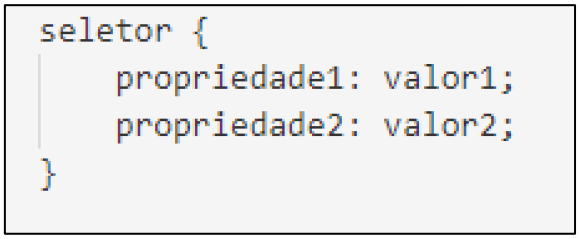
O CSS é utilizado para definir a apresentação de documentos escritos em uma linguagem de marcação, como *Hypertext Markup Language* (HTML) ou *Extensible Markup Language* (XML). Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento. Para saber melhor sobre como funciona o CSS e sua estrutura temos que falar dos seletores.

Um seletor representa uma estrutura. Essa estrutura é usada como uma condição para determinar quais elementos de um grupo de elementos serão formatados.

Seletores encadeados e seletores agrupados são a base do CSS. Você aprende naturalmente durante o uso do dia a dia (FERREIRA; EIS, 2011, p. 139).

Exemplo de seletores:

Figura 4 – Exemplo de Seletores CSS



Fonte: Autoria própria, 2023.

De acordo com Caelum (2023), quando queremos estilizar elementos específicos é melhor utilizar o atributo *class*="" O comportamento no CSS será idêntico ao do atributo *id*="", mas *class* foi feito para ser usado no CSS e no JavaScript.

Segundo Duckett (2016), o conceito das propriedades utilizadas:

- *Color*: propriedade que permite a especificação da cor do texto do elemento. Podendo ser definida em valores *Red, Green, Blue* (RGB), códigos hexadecimais ou nome das cores.
- *Font-size*: define o tamanho da fonte do texto, podendo ser definida por pontos, pixels (PX) ou porcentagens, que são as maneiras mais comuns.
- *Margin*: define o espaçamento entre as caixas. Se uma sobrepor a outra, a margem maior será usada e a menor desconsiderada.
- *Width* e *height*: define a largura e altura da caixa. Podendo ser definida por porcentagem ou pixels.
- *Background*: abreviação para todos os tipos de *background*, configura o fundo do seu elemento. Deve ser definido de acordo com a seguinte ordem: *background-color*, que determina a cor do fundo; *background-image*, que posiciona uma imagem atrás de qualquer elemento ao se colocar o caminho da imagem entre parênteses e aspas; *background-repeat*, que repete a imagem colocada no fundo, vertical e horizontalmente; *background-attachment*, que determina se a imagem do fundo ficará fixa ou irá se mover quando o usuário rolar a página; e *background-position*, para quando a

imagem não se repetir, é utilizado para indicar ao navegador onde a imagem será posicionada. *Background-size* é utilizado para definir a dimensão da imagem do fundo.

- *Padding*: é a margem interna de um elemento, definindo o espaço entre o conteúdo e a borda do elemento. O valor pode ser especificado em *pixels*, em porcentagem ou em outras unidades de medida.
- *Font-family*: define a família da fonte utilizada no texto de um elemento.
- *Text-align*: define o alinhamento do texto de um elemento, podendo ser "*left*" (esquerda), "*center*" (centro), "*right*" (direita) ou "*justify*" (justificado).
- *Border*: define a borda de um elemento, incluindo sua largura, estilo e cor. O valor pode ser especificado separadamente para cada lado da borda (*top, right, bottom, left*).
- *Position*: define como um elemento deve ser posicionado em relação ao seu container. O valor "*absolute*" faz com que um elemento seja posicionado em relação ao elemento pai mais próximo com a posição definida, enquanto o valor "*fixed*" faz com que um elemento seja posicionado em relação à janela de visualização.
- *Z-index*: define a ordem de empilhamento dos elementos. Um elemento com um valor de *z-index* maior ficará sobre um elemento com um valor de *z-index* menor.
- *Box-shadow*: adiciona uma sombra ao redor de um elemento, permitindo definir o tamanho, cor e tipo de sombra.
- *Overflow*: define o comportamento do conteúdo que transborda os limites de um elemento.

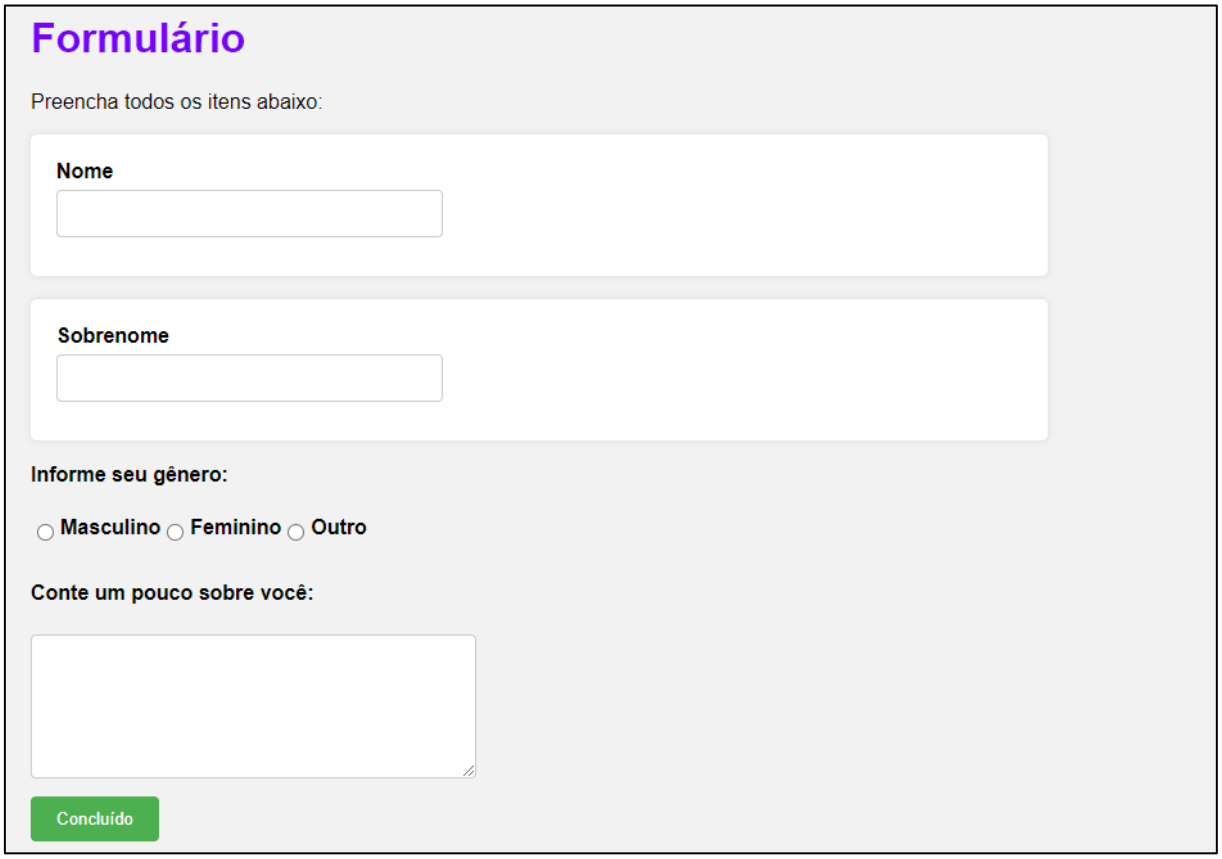
A seguir exemplo da aplicação de CSS no projeto apresentado mais cedo:

Figura 5 – Código em CSS

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    padding: 20px;
    margin: 0;
}
h1 {
    color: #7700ff;
}
form {
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
}
label {
    display: block;
    font-weight: bold;
    margin-bottom: 5px;
}
input[type="text"],
textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}
input[type="radio"] {
    margin-right: 5px;
}
button {
    padding: 10px 20px;
    background-color: #4caf50;
    color: #fff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
button:hover {
    background-color: #45a049;
}
```

Fonte: Autoria própria, 2023.

Figura 6 – Código em CSS rodando no navegador



Fonte: Autoria própria, 2023.

2.3 Bootstrap

O Bootstrap é o mais popular *framework* JavaScript, HTML e CSS para desenvolvimento de sites e aplicações *web* responsivas e alinhadas com a filosofia *mobile first*. Torna o desenvolvimento *front-end* muito mais rápido e fácil. Indicado para desenvolvedores de todos os níveis de conhecimento, dispositivos de todos os tipos e projetos de todos os tamanhos. (SAMY, 2015, p. 20).

No dia 19 de agosto de 2011, Mark Otto, um desenvolvedor trabalhando no Twitter, criador do Bootstrap juntamente com Jacob Thornton, anunciou ao mundo o lançamento do Bootstrap em um artigo publicado no blog do Twitter. Iniciou o artigo dizendo estar feliz com o lançamento de um kit de ferramentas *front-end* destinado a agilizar o desenvolvimento de aplicações *web*. Naquele artigo, relatou que no início do Twitter cada desenvolvedor usava a biblioteca que lhe era familiar para solucionar os problemas de *front-end*. Esse procedimento criou inconsistências, dificultando a integração, a escalabilidade e a manutenção das aplicações criadas por diferentes

desenvolvedores da equipe. Bootstrap foi a proposta de solução para aquelas inconsistências. Apresentada na primeira Twitter *Hackweek* realizada na semana de 22 a 29 de outubro de 2011, de lá saiu com sua primeira versão estável e pronta para uso. (SAMY, 2015, p. 21).

De acordo com Maurício (2015), assim o *template* mínimo para criar uma aplicação que use o Bootstrap é mostrado a seguir:

Figura 7 – Template de aplicação de Bootstrap

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo Bootstrap</title>
    <!--Importando arquivos CSS e JS do bootstrap-->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>

</body>
</html>
```

Fonte: (Bootstrap 3.3.5, 2015)

A seguir um exemplo de uso de classes do *Bootstrap*:

Figura 8 – Propriedades das classes do Bootstrap

```
.row {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -ms-flex-wrap: wrap;
  flex-wrap: wrap;
  margin-right: -15px;
  margin-left: -15px;
}
.col {
  -ms-flex-preferred-size: 0;
  flex-basis: 0;
  -webkit-box-flex: 1;
  -ms-flex-positive: 1;
  flex-grow: 1;
  max-width: 100%;
}
.text-center {
  text-align: center !important;
}
.container {
  min-width: 992px !important;
}
```

Fonte: Autoria própria, 2023.

Figura 9 – Representação de classes do Bootstrap no HTML

Fonte: (Bootstrap, 2023)

2.4 Javascript

O Javascript foi criado na Netscape na fase inicial da *web* e, tecnicamente, "Javascript" é marca registra da licenciada pela Sun Microsystems (agora Oracle), usada para descrever a implementação da linguagem pelo Netscape (agora Mozilla). A Netscape enviou a linguagem para a *European Computer Manufacturer's Association* (ECMA) para padronização e, devido a questões relacionadas à marca registrada, a versão padronizada manteve o nome estranho "ECMAScript". Pelos mesmos motivos ligados à marca registrada, a versão da Microsoft da linguagem é formalmente conhecida como "JScript". Na prática, quase todo mundo chama a linguagem de Javascript. (FLANAGAN, 2013, p. 2).

De acordo com Preston (2016), você pode usar o Javascript para ajudar os usuários a interagirem melhor com páginas *web*. O Javascript também pode ser usado para controlar o navegador, comunicar-se de forma assíncrona com o servidor, alterar de forma dinâmica o conteúdo de uma página *web* e para desenvolver jogos e aplicações móveis e de *desktop*.

Segundo Michael (2008), é preciso informar ao navegador que será usado o Javascript por meio da *tag* <script>, colocando como atributo o tipo de linguagem que será utilizada, e então a *tag* de abertura ficaria <script type=”text/javascript”>.

Ela pode ser colocada em qualquer lugar do código HTML, mas geralmente fica dentro do elemento <head>.

Existe uma *Application Programming Interface* (API) de console simples, implementada de forma portável pelos navegadores modernos. Você pode usar a função *console.log()* para exibir texto na console. Isso muitas vezes é surpreendentemente útil ao se fazer depuração, sendo que alguns exemplos deste livro (mesmo na seção de linguagem básica) utilizam *console.log()* para produzir saída simples. (FLANAGAN, 2013, p. 3).

Segue abaixo um exemplo do uso do *console.log()*:

Figura 10 – Exemplo de uso de console.log

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <title>Exemplo</title>
</head>
<body>
    <script type="text/javascript">
        console.log("Digite a saida aqui")
    </script>
</body>
</html>
```

Fonte: Autoria própria, 2023.

Uma maneira semelhante, porém mais invasiva, de exibir saída ou mensagens de depuração é passar uma *string* de texto para a função *alert()*, a qual as exibe em uma caixa de diálogo modal (FLANAGAN, 2013, p. 3).

A seguir um exemplo da função *alert()*:

Figura 11 – Exemplo da função Alert

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <title>Exemplo</title>
</head>
<body>
    <script type="text/javascript">
        alert("Mensagem de alerta")
    </script>
</body>
</html>
```

Fonte: Autoria própria, 2023.

Indo para algo mais profundo, mas seguindo o básico, será mostrado alguns exemplos de *tags*/componentes comuns no Javascript:

- *var*: Palavra-chave usada para declarar uma variável em Javascript, permitindo armazenar e manipular valores durante a execução do programa.
- *document.getElementById()*: Método que permite selecionar um elemento específico do DOM (Modelo de Objeto de Documento) com base no seu atributo "id". Retorna uma referência ao elemento encontrado.
- *value*: Propriedade que contém o valor atual de um elemento de entrada, como um campo de texto ou uma caixa de seleção. Permite acessar ou definir o valor do elemento.
- *console.log()*: Função usada para exibir informações no console do navegador durante o desenvolvimento, permitindo imprimir valores, variáveis ou mensagens para depurar e verificar o estado do código.
- Concatenação: Processo de unir duas ou mais *strings* em Javascript. Pode ser realizado utilizando o operador de adição (+) para combinar as *strings* e criar uma *string* resultante.

A seguir um exemplo de aplicação de Javascript:

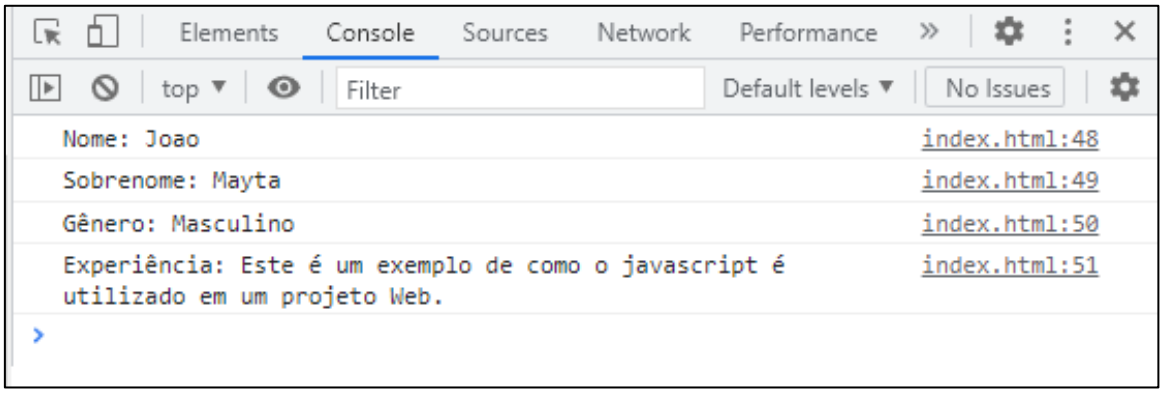
Figura 12 – Código em Javascript do formulário

```
<script>
    document.getElementById("myForm").addEventListener("submit", function (event) {
        event.preventDefault();

        var nome = document.getElementById("nome").value;
        var sobrenome = document.getElementById("sobrenome").value;
        var genero = document.querySelector('input[name="genero"]:checked').value;
        var experiencia = document.getElementById("experiencia").value;

        console.log("Nome: " + nome);
        console.log("Sobrenome: " + sobrenome);
        console.log("Gênero: " + genero);
        console.log("Experiência: " + experiencia);
    });
</script>
```

Fonte: Autoria Própria, 2023.

Depois de informar os dados requisitados no site, ele mostra os dados colocados no console, como segue o exemplo a seguir:

Figura 13 – Console formulário



Fonte: Autoria própria, 2023.

2.5 PHP

A linguagem de programação PHP, que no início significava *Personal Home Page Tools,* porém mudou para *Hypertext Preprocessor*, foi criada no outono de 1994 por Rasmus Lerdorf. Essa linguagem era formada por um conjunto de *scripts* escritos em linguagem C, voltados à criação de páginas dinâmicas que Rasmus utilizava para monitorar o acesso ao seu currículo na internet. Com o tempo, mais pessoas passaram a utilizá-la e Rasmus adicionou vários recursos, como a interação com bancos de dados. Em 1995, o código-fonte do PHP foi liberado, e com isso mais

desenvolvedores puderam se juntar ao projeto. Naquela época, por um breve período (DALL'OGLIO, 2016, p. 21).

O PHP passou por várias reescritas de código ao longo do tempo e nunca parou de conquistar novos adeptos. Uma segunda versão foi lançada em novembro de 1997, sob o nome PHP 2. Naquele momento, aproximadamente 60 mil domínios, ou 1% da internet, já utilizavam PHP, que era mantido principalmente por Rasmus. (DALL'OGLIO, 2016, p. 21).

Segundo Pablo (2016), ao longo de mais de uma década, o PHP vem adicionando mais e mais recursos e se consolida ano após ano como uma das linguagens de programação orientadas a objetos que mais crescem no mundo. Estima-se que o PHP seja utilizado em mais de 80% dos servidores *web* existentes, tornando-a disparadamente a linguagem mais utilizada para desenvolvimento *web*. Ao longo do livro, veremos esses recursos empregados em exemplos práticos.

Segundo Pablo (2016), algumas *tags* possuem seus significados, deixamos aqui as utilizadas em qualquer estrutura:

- <?php: indica o início de um trecho de código PHP.
- *echo*: serve para escrever algo na tela.
- ?>: indica o término de um trecho de código PHP.

A seguir a estrutura de um PHP:

Figura 14 – Estrutura de um PHP

```
<!DOCTYPE html>
<html lang="pt-br">
    <body>
        <?php
            echo "Estrutura PHP"
        ?>
    </body>
</html>
```

Fonte: Autoria própria, 2023.

De acordo com Tim (2003), PHP permite incorporar fragmentos de código em páginas HTML normais-código esse que é interpretado à medida que suas páginas são oferecidas aos usuários. O PHP também serve como uma linguagem de "cola",

facilitando a conexão de suas páginas *web* com o banco de dados do lado do servidor.

A seguir um exemplo mais estruturado de um PHP, trazendo declarações de variáveis com fragmentos de *tags* HTML:

Figura 15 – Exemplo de declaração em PHP

```php
<?php
    $nome = "João";
    $idade = 25;

    function exibirMensagem($nome, $idade) {
        echo "<h1>Bem-vindo, $nome!</h1>";
        echo "<p>Sua idade é: $idade anos.</p>";
    }
?>
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de PHP com HTML</title>
</head>
<body>
    <?php
        // Chamando a função exibirMensagem() definida acima e passando as variáveis como argumentos
        exibirMensagem($nome, $idade);
    ?>
</body>
</html>
```

Fonte: Autoria própria, 2023.

Figura 16 – Exemplo PHP

Fonte: Autoria própria, 2023.

2.6 Banco de dados

Banco de dados ou base de dados pode simplesmente ser definida como um local – físico ou virtual – onde se armazenam dados de forma organizada e indexada. Percebam que não estamos nem na parte de informática e sim na parte física e

analógica. Com isso em mente, podemos citar como exemplo de banco de dados, o bloco de notas, o documento no *word*, uma planilha, um caderno e até mesmo um baú. A internet também pode ser considerada um banco de dados quando consultamos qualquer bibliografia de banco de dados (PERA, 2021, p. 9).

Segundo Bruno (2021), o vocabulário utilizado pelos autores são os seguintes:

- Entidades: São as tabelas do banco de dados;
- Atributos: São os campos das entidades;
- Domínios: São todos os valores possíveis de dados armazenados em um atributo.
- Tupla: Cada registro armazenado numa entidade; lembre-se que um registro se refere ao conjunto de dados armazenados em todas os atributos de uma tabela.
- Chave primária: um ou mais atributos, usados para identificar e unificar as tuplas armazenadas.
- Chave estrangeira: atributo existente em uma entidade filho, ou seja, entidade que recebe dados de outra entidade em um relacionamento. A entidade FILHO, no relacionamento, deve possuir os mesmos atributos da chave primária da tabela PAI, e estes devem ser do mesmo tipo e tamanho, a fim de tornar possível o relacionamento entre as tabelas.

Um Sistemas gerenciadores de banco de dados (SGBD), é um software responsável por tornar o banco de dados gerenciável permitir que ele seja manipulado, ele ainda não é o software que vai ao usuário final e sim o que o DBA (Administrador de banco de dados) ou *Database Admin* irá utilizar. Sua principal função é facilitar a interface com o banco e do DBA (PERA, 2021, p. 10).

Segundo William (2020), com um sistema de banco dados, os aplicativos não têm qualquer conhecimento dos mecanismos relacionados com as operações de gravação e leitura física dos dados. O que eles fazem é simplesmente se comunicar com o *software* de gerenciamento para recuperar ou armazenar as informações desejadas. Desta forma, diversos programas podem acessar um mesmo banco de dados e qualquer alteração em sua estrutura não pressupõe, necessariamente, modificações nos aplicativos.

2.6.1 NoSQL

De acordo com Bruno (2021), um erro muito comum é quando dizem que os bancos de dados não relacionais não armazenam bem dados de relacionamento. Eles podem armazenar dados de relacionamento, mas apenas os armazenam de maneira diferente dos bancos de dados relacionais. Muitos consideram a modelagem relacionamentos nos bancos de dados *Not Only SQL* (NoSQL), mais fácil do que nos bancos de dados SQL, porque os dados relacionados não precisam ser divididos entre as tabelas. Os modelos de dados NoSQL permitem por exemplo, que os dados relacionados sejam feitos em uma única estrutura de dados. Diferentemente dos bancos relacionais, a estrutura de dados não precisa ser definida previamente, portanto, em uma mesma “tabela” você pode ter dados com propriedades diferentes. Os bancos de dados NoSQL surgiram no final dos anos 2000, à medida que o custo do armazenamento diminuiu drasticamente. Já se foram os dias em que era necessário criar um modelo de dados complexo e difícil de gerenciar, simplesmente com o objetivo de reduzir a duplicação de dados.

2.6.2 Firebase

O *Firebase* é uma plataforma de desenvolvimento de aplicativos móveis e *web*, que foi adquirida pelo Google em 2014. Ele oferece uma série de serviços integrados, como banco de dados em tempo real, autenticação de usuários, armazenamento em nuvem, hospedagem de aplicativos e serviços de mensagens. O *Firebase* é amplamente utilizado por desenvolvedores de aplicativos móveis e *web* em todo o mundo, devido à sua facilidade de uso, escalabilidade e alta disponibilidade. (Google, 2022).

Um dos principais benefícios do *Firebase* é que ele permite que os desenvolvedores criem aplicativos de alta qualidade e com recursos avançados, sem a necessidade de escrever código complexo. Além disso, ele possui uma interface intuitiva, que permite que os desenvolvedores gerenciem facilmente seus aplicativos e serviços (Google, 2023).

Um recurso importante do *Firebase* é o seu banco de dados em tempo real, que permite que os aplicativos sincronizem dados em tempo real, sem a necessidade de atualizações manuais. Isso torna o *Firebase* ideal para aplicativos que requerem

atualizações em tempo real, como aplicativos de mensagens instantâneas e jogos *multiplayer* (Google, 2023).

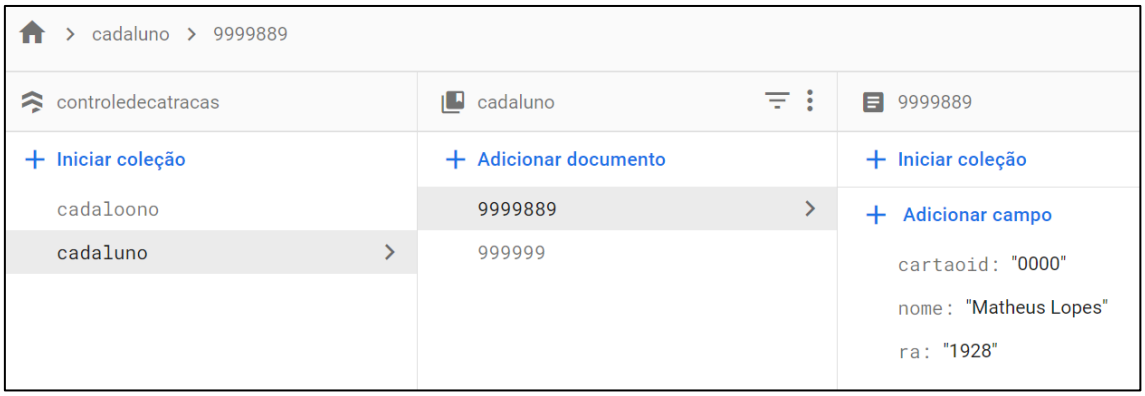
De acordo com a Google (2023), o Firebase é altamente personalizável e pode ser facilmente integrado a outras plataformas e ferramentas de desenvolvimento. Ele oferece uma série de bibliotecas de código aberto, que permitem que os desenvolvedores personalizem seus aplicativos e adicionem recursos avançados.

Com o Firebase é possível criar aplicativos poderosos, seguros e escalonáveis, desta forma, diversos são os seus serviços disponíveis, como:

- *Cloud Firestore:* Oferece sincronização ao vivo e suporte *offline*, além de consultas eficientes a dados;
- *Cloud Functions:* Permite criar lógicas personalizadas que serão executadas nos aplicativos conectados ao *firebase*;
- *Authentication:* Com o *Firebase Auth* é possível gerenciar seus usuários de maneira simples e segura, oferecendo métodos de autenticação e autorização;
- *Hosting:* Permite hospedar HTML, CSS e JavaScript para seu site, além de outros ativos fornecidos pelo desenvolvedor, como gráficos, fontes e ícones;
- *Cloud Storage:* Recurso que permite o armazenamento de arquivos na nuvem para que sejam compartilhados entre os aplicativos;
- *Realtime Database*: Eficiente e de baixa latência para aplicativos para dispositivos móveis, o *Realtime Database* é um banco de dados com atualização em tempo real, permitindo o compartilhamento de informação entre diversos usuários de um app instantaneamente.

Segue abaixo um exemplo de utilização da ferramenta *Firestore* com o intuito de simular um controle de catracas:

Figura 17 – Representação de Firestore no Firebase



Fonte: Autoria própria, 2023.

2.7 Inteligência Artificial

A Inteligência Artificial (IA), é um dos campos mais recentes em ciências e engenharia. O trabalho começou logo após a Segunda Guerra Mundial, e o próprio nome foi cunhado em 1956. Juntamente com a biologia molecular, a IA é citada regularmente como “o campo em que eu mais gostaria de estar” por cientistas de outras disciplinas. Um aluno de física pode argumentar, com boa dose de razão, que todas as boas ideias já foram desenvolvidas por Galileu, Newton, Einstein e o resto. IA, por outro lado, ainda tem espaço para vários Einsteins e Edisons em tempo integral. Atualmente, a IA abrange uma enorme variedade de subcampos, do geral (aprendizagem e percepção) até tarefas específicas, como jogos de xadrez, demonstração de teoremas matemáticos, criação de poesia, direção de um carro em estrada movimentada e diagnóstico de doenças. A IA é relevante para qualquer tarefa intelectual; é verdadeiramente um campo universal. (NORVIG; RUSSEL, 2013, p. 24).

De acordo com Dennis (2010), a inteligência artificial surgiu na década de 50 com o objetivo de desenvolver sistemas para realizar tarefas que, no momento são melhor realizadas por seres humanos que por máquinas, ou não possuem solução algorítmica viável pela computação convencional.

2.7.1 OpenAI

A *Application Programming Interface* (API), ou em português Interface de Programação de Aplicação, OpenAI pode ser aplicada a praticamente qualquer tarefa que exija compreensão ou geração de código e linguagem natural. A API OpenAI também pode ser usada para gerar e editar imagens ou converter fala em texto. Oferecemos uma variedade de modelos com diferentes capacidades e preços, bem como a capacidade de ajustar modelos personalizados. (OpenAI, 2023).

Os modelos Transformadores Pré-treinados Generativos (GPT), da OpenAI foram treinados para entender a linguagem e o código naturais. GPTs fornecem saídas de texto em resposta às suas entradas. As entradas para GPTs também são chamadas de "*prompts*". Projetar um *prompt* é essencialmente como você “programa” um modelo GPT, geralmente fornecendo instruções ou alguns exemplos de como concluir uma tarefa com êxito. Os GPTs podem ser usados em uma grande variedade de tarefas, incluindo geração de conteúdo ou código, resumo, conversação, redação criativa e muito mais. (OpenAI, 2023).

Usando GPTs, você pode criar aplicativos para:

- Rascunhos de documentos.
- Escrever código de computador.
- Responder a perguntas sobre uma base de conhecimento.
- Analisar textos.
- Criar agentes de conversação.
- Dê ao software uma interface de linguagem natural.
- Tutora em diversas disciplinas.
- Traduzir idiomas.
- Simular personagens para jogos.

2.8 UML

Unified Modeling Language (UML), é uma linguagem-padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software. A UML é adequada para a modelagem de sistemas, cuja abrangência poderá incluir sistemas de informação corporativos a serem distribuídos a aplicações baseadas na Web e até sistemas complexos embutidos de tempo real. É uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas. Apesar de sua expressividade, não é difícil compreender e usar a UML. Aprender a aplicar a UML de maneira efetiva tem início com a formação de um modelo conceitual da linguagem, o que pressupõe o entendimento de três principais elementos: os blocos básicos de construção da UML, as regras que determinam como esses blocos de construção deverão ser combinadas e alguns mecanismos básicos que se aplicam a toda a linguagem. A UML é apenas uma linguagem e, portanto, é somente uma parte de um método para desenvolvimento de software. A UML é independente do processo, apesar de ser perfeitamente utilizada em processo orientado a casos de usos, centrado na arquitetura, iterativo e incremental. (BOOCH, 2012, p. 46).

De acordo com Grady (2012), a UML é uma linguagem destinada a:

- Visualizar
- Especificar
- Construir
- Documentar os artefatos de um sistema complexo de *software*

> A UML é uma linguagem para construção; não é uma linguagem visual de programação, mas seus modelos podem ser diretamente conectados a várias linguagens de programação. Isso significa que é possível mapear os modelos da UML em linguagens de programação tais como Java, C++, Visual Basic ou até tabelas de bancos de dados relacionais ou o armazenamento de dados persistentes de um banco de dados orientado a objetos. A UML é capaz de representar tudo que possa ser mais bem expresso em termos gráficos, enquanto as linguagens de programação representam o que é mais bem expresso em termos textuais. (BOOCH, 2012, p. 49).

2.8.1 Diagrama de Caso de Uso

Um caso de uso especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de sequências de ações, incluindo variantes realizadas pelo sistema para produzir um resultado observável do valor de um ator. (BOOCH, 2012, p. 343).

De acordo com Grady (2012), os casos de usos podem ser aplicados para captar o comportamento pretendido do sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento é implementado. Os casos de uso fornecem uma maneira para os desenvolvedores chegarem a uma compreensão comum com os usuários finais do sistema e com os especialistas do domínio. Além disso, os casos de uso servem para ajudar a validar a arquitetura e para verificar o sistema à medida que ele evolui durante seu desenvolvimento. À proporção que você implementa o seu sistema, esses casos de uso são realizados por colaborações cujos elementos trabalham em conjunto para a execução de cada caso de uso. Casos de uso bem estruturados denotam somente o comportamento essencial do sistema ou subsistema e não são amplamente gerais, nem muito específicos.

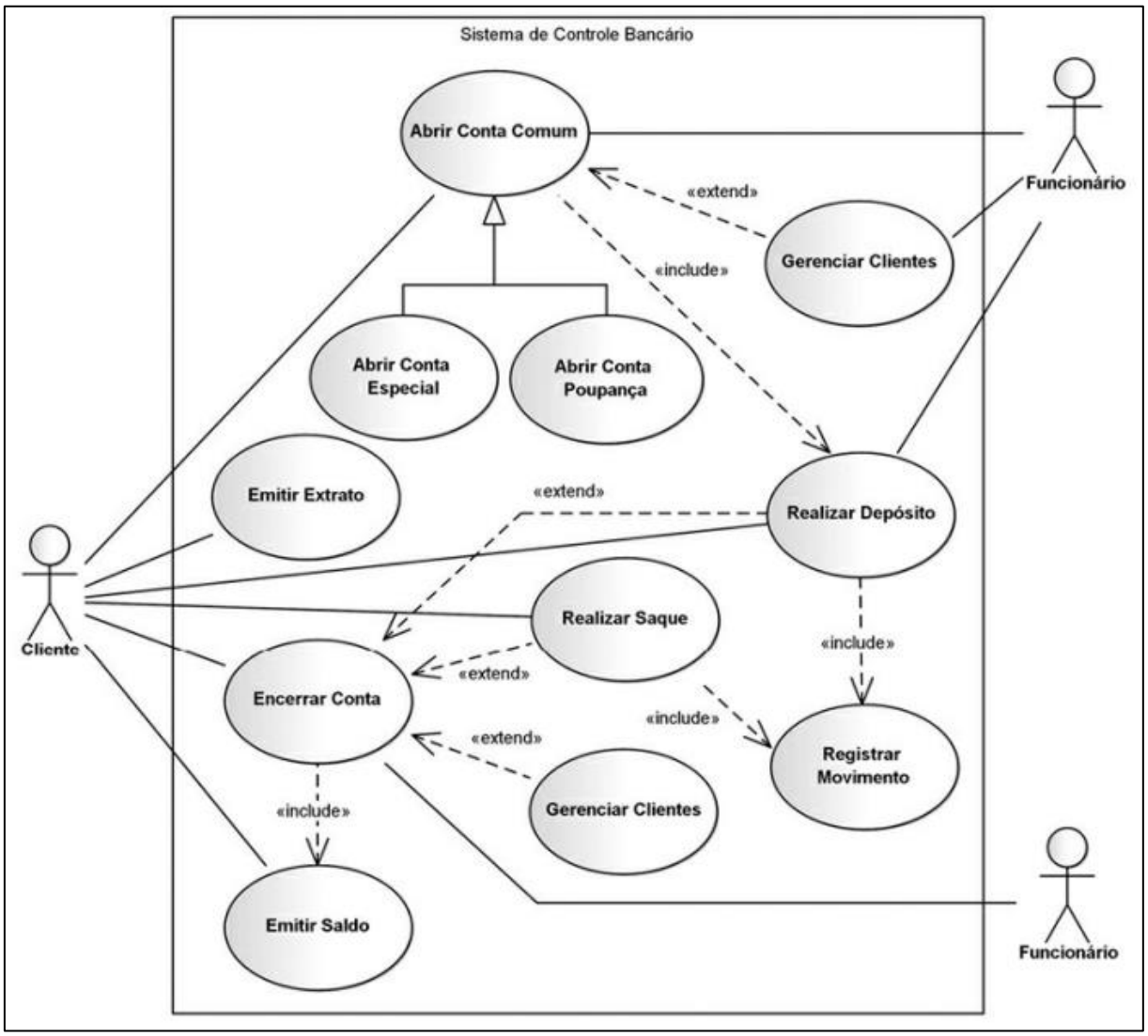
Conforme toda criação de diagrama, é possível detectar suas principais características, segue algumas abaixo:

- Interação entre Ator e Sistema: Os casos de uso descrevem como os atores humanos ou autômatos interagem com um sistema para alcançar um objetivo específico. Os atores esperam que o sistema se comporte de acordo com as maneiras previstas.
- Descrição de Sequências de Ações: Um caso de uso é uma descrição de um conjunto de sequências de ações realizadas pelo sistema para produzir um resultado observável de valor para o ator. Ele especifica o comportamento do sistema ou de uma parte do sistema.
- Foco no Comportamento Essencial: Casos de uso bem estruturados descrevem apenas o comportamento essencial do sistema ou subsistema, evitando serem muito genéricos ou muito específicos.

- Especificação Independente de Implementação: Os casos de uso podem ser especificados sem detalhar como serão implementados. Eles se concentram no comportamento desejado do sistema, permitindo uma comunicação mais fácil entre usuários finais, especialistas do domínio e desenvolvedores.
- Modelagem em UML: Na UML, os casos de uso são modelados como descrições de sequências de ações, representando a interação entre atores externos ao sistema e as principais abstrações do sistema. Eles são usados para visualizar, especificar, construir e documentar o comportamento pretendido do sistema durante a captação e análise de requisitos.

Segue um exemplo de um caso de uso no sistema de saque de conta em um banco:

Figura 18 – Representação do diagrama de Caso de Uso



Fonte: (UML 2 – Uma abordagem prática, 2018).

2.8.2 Diagrama de Atividade

Os diagramas de atividades serão empregados para fazer a modelagem de aspectos dinâmicos do sistema. Na maior parte, isso envolve a modelagem das etapas sequenciais (e possivelmente concorrentes) em um processo computacional. Com um diagrama de atividade, você também pode fazer a modelagem do fluxo de um objeto, à medida que ele passa de um estado para outro em pontos diferentes do fluxo de controle. Os diagramas de atividades poderão permanecer isolados para visualizar, especificar, construir e documentar a dinâmica de uma sociedade de objetos, ou poderão ser utilizados para fazer a modelagem do fluxo de controle de uma operação. (BOOCH, 2012, p. 402).

Segundo Grady (2012), o diagrama de atividades é apenas um tipo especial de diagrama e compartilha as mesmas propriedades comuns de todos os demais diagramas – um nome e um conteúdo gráfico que são a projeção em um modelo. O que diferencia um diagrama de interação dos demais tipos de diagramas é o seu conteúdo particular.

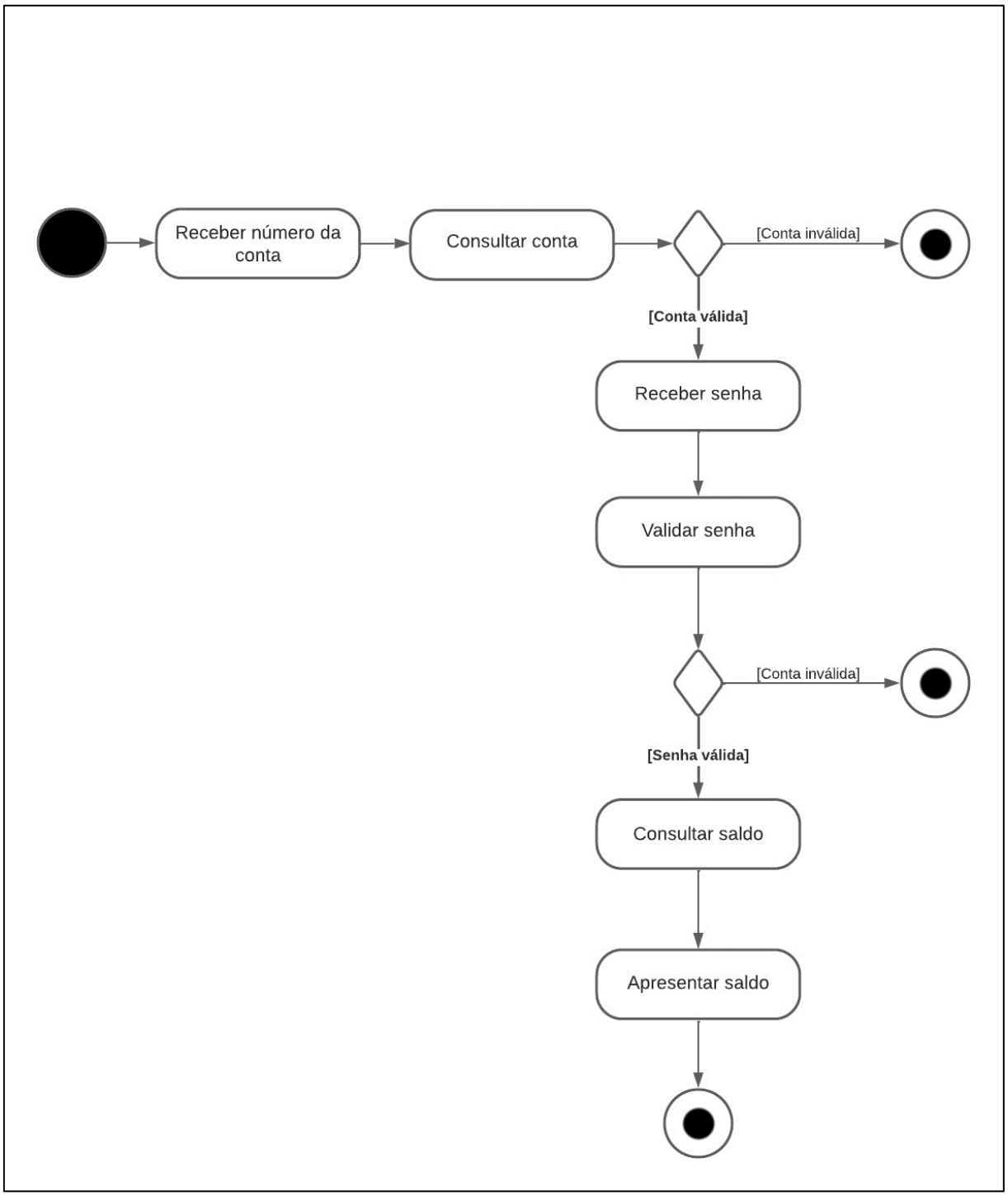
Segue a baixo separado em tópicos, as principais características do diagrama de atividade:

- Aspectos Dinâmicos dos Sistemas: Os diagramas de atividades são uma das cinco opções disponíveis na UML para modelar os aspectos dinâmicos de sistemas. Eles representam graficamente o fluxo de controle de uma atividade para outra, levando em consideração tanto a concorrência quanto as ramificações de controle.
- Comparação com Outros Diagramas: Além dos diagramas de atividades, existem outros diagramas na UML para modelar aspectos dinâmicos, como os diagramas de sequências, comunicação, gráficos de estados e casos de uso. Cada um deles enfoca diferentes aspectos da dinâmica do sistema.
- Modelagem de Etapas Sequenciais e Concorrentes: Os diagramas de atividades são usados principalmente para modelar etapas sequenciais e, possivelmente, concorrentes em um processo computacional. Eles também podem ser usados para modelar o fluxo de um objeto à medida que ele transita de um estado para outro no fluxo de controle.

- Ênfase no Fluxo de Controle: Enquanto os diagramas de interação destacam o fluxo de controle de um objeto para outro, os diagramas de atividades enfatizam o fluxo de controle de uma etapa para outra. Uma atividade representa uma execução estruturada em andamento de um comportamento, composta por ações individuais que podem alterar o estado do sistema ou comunicar mensagens.
- Importância na Modelagem Dinâmica e Construção de Sistemas Executáveis: Os diagramas de atividades não apenas são úteis para modelar aspectos dinâmicos de um sistema, mas também são empregados na construção de sistemas executáveis por meio de engenharia direta e reversa.
- Exemplo de Fluxo de Trabalho: Um exemplo de aplicação dos diagramas de atividades é a modelagem do fluxo de trabalho associado à construção de uma casa. Diversas atividades paralelas, condições, ramificações e loops são considerados nesse contexto.
- Analogia com Gráficos de Gantt e Pert: Assim como os gráficos de Gantt e Pert são usados na construção para visualizar, especificar, construir e documentar o fluxo de trabalho de um projeto, os diagramas de atividades desempenham um papel semelhante na modelagem de sistemas complexos de software.

Segue abaixo um exemplo de consulta e saque de dinheiro em um banco:

Figura 19 – Representação do diagrama de Atividade



Fonte: Autoria própria, 2023.

2.8.3 Diagrama de Objetos

Os diagramas de objetos fazem a modelagem de instâncias de itens contidos em diagramas de classes. Um diagrama de objetos mostra um conjunto de objetos e seus relacionamentos em determinado ponto no tempo. (BOOCH, 2012, p. 289).

De acordo com Grady (2012), você usa os diagramas de objetos para fazer a modelagem da visão estática do projeto ou do processo de um sistema. Isso envolverá a modelagem de um retrato do sistema em determinado momento e a representação de um conjunto de objetos, seus estados e relacionamentos. Os diagramas de objetos não são importantes apenas para a visualização, especificação e documentação de modelos estruturais, mas também para a construção de aspectos estáticos de sistemas por meio de engenharia direta e reversa.

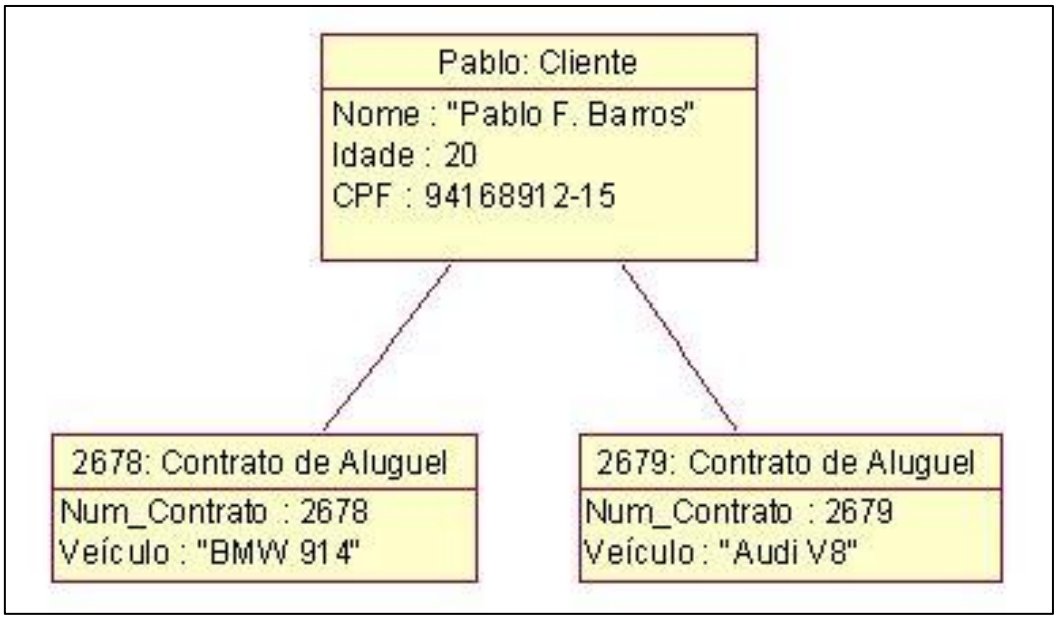
Segue separado em alguns tópicos das principais características de um diagrama de objetos:

- Modelagem de Instâncias: Os Diagramas de Objetos permitem a modelagem de instâncias específicas de objetos em um determinado ponto no tempo. Eles representam um conjunto de objetos e seus relacionamentos.
- Visão Estática: Os Diagramas de Objetos são usados para modelar a visão estática de um sistema, retratando um instantâneo do sistema em um determinado momento. Eles representam objetos, seus estados e relacionamentos.
- Construção e Documentação: Além de visualização e especificação, os Diagramas de Objetos são úteis na construção e documentação de aspectos estáticos de sistemas complexos de software. Eles auxiliam na engenharia direta e reversa do sistema.
- Compreensão dos Relacionamentos: Assim como no futebol americano, onde a observação individual dos jogadores não revela a estrutura e estratégia do jogo, os Diagramas de Objetos permitem compreender os relacionamentos entre os objetos e como eles colaboram entre si.

- Congelar um Momento no Tempo: Os Diagramas de Objetos capturam um instantâneo do sistema em um determinado momento, fornecendo uma representação estática das instâncias de objetos e seus relacionamentos.
- Identificação de Conexões Quebradas: Nos sistemas orientados a objetos, problemas muitas vezes surgem devido a conexões quebradas entre objetos ou estados danificados. Os Diagramas de Objetos ajudam a identificar essas falhas.
- Complemento aos Diagramas de Classes: Os Diagramas de Objetos são parte da UML e complementam os Diagramas de Classes. Enquanto os diagramas de classes representam a estrutura do sistema, os diagramas de objetos mostram instâncias específicas dos itens encontrados nos diagramas de classes.

A Figura abaixo apresenta um exemplo de diagrama de objetos, onde um cliente aluga um carro, emitindo o contrato do aluguel com suas principais informações:

Figura 20 – Representação do diagrama de Objetos



Fonte: (Eu faço Programas, 2018)

2.8.4 Diagrama de Sequência

Um diagrama de sequência é um diagrama de interação que dá ênfase à ordenação temporal de mensagens. Um diagrama de sequência mostra um conjunto de papéis e as mensagens enviadas e recebidas pelas instâncias que representam os papéis. Use os diagramas de sequência para ilustrar a visão dinâmica de um sistema. (BOOCH, 2012, p. 162).

Segundo Grady (2012), durante uma interação, um objeto tipicamente muda os valores de seus atributos, seu estado ou seus papéis. Você pode representar as modificações de um objeto em um diagrama de sequência, mostrando o estado ou os valores da linha de vida. Em um diagrama de sequência, o tempo de vida, a criação e a destruição de objetos ou papéis são explicitamente apresentadas pela extensão vertical de suas linhas de vida.

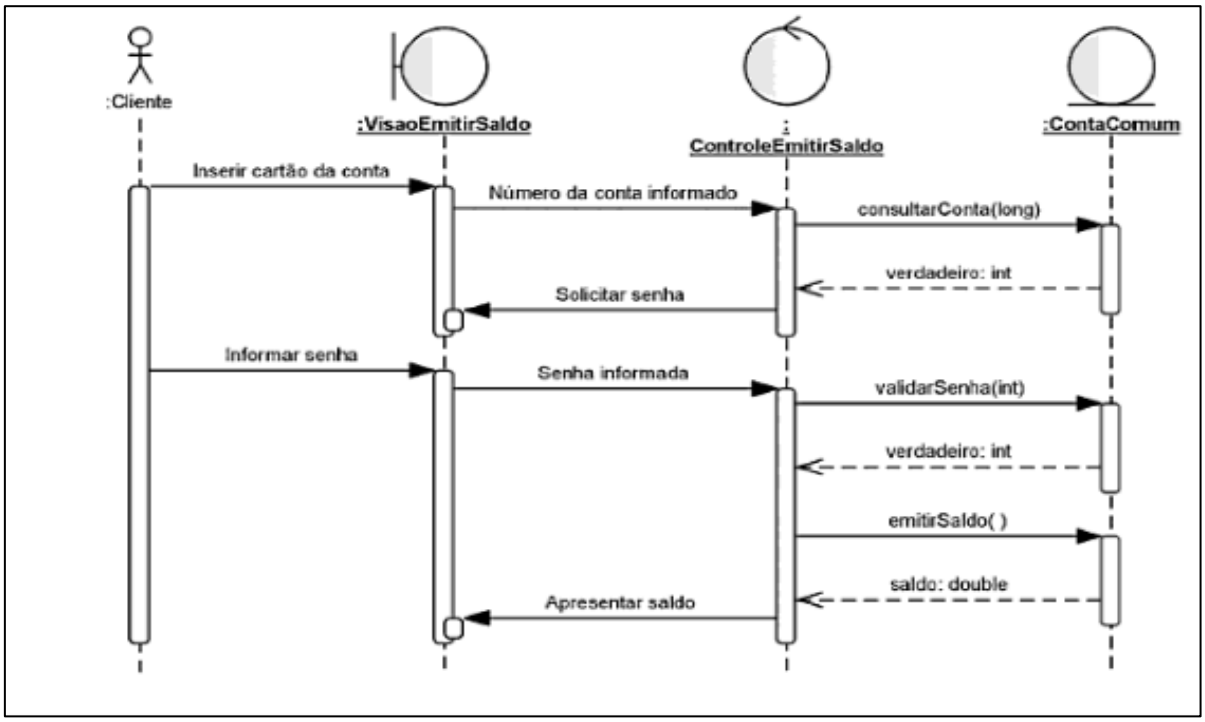
Segue as abaixo algumas características do diagrama de sequência:

- Ênfase na Ordenação Temporal: Um diagrama de sequência destaca a ordenação temporal das mensagens em uma interação. Os objetos participantes são colocados no eixo X, com o objeto que inicia a interação à esquerda e os objetos subordinados crescendo à direita. As mensagens trocadas entre os objetos são representadas ao longo do eixo Y, em ordem crescente de tempo.
- Linha de Vida do Objeto: Cada objeto tem uma linha de vida vertical, representando sua existência ao longo do tempo. A linha de vida se estende verticalmente no diagrama. Alguns objetos têm duração igual à da interação e são alinhados na parte superior do diagrama. Outros objetos podem ser criados ou destruídos durante a interação, indicados por estereótipos "create" e "destroy".
- Papéis Prototípicos: Em interações prototípicas, as linhas de vida não representam objetos específicos, mas sim papéis prototípicos que podem ser assumidos por diferentes objetos em cada instância da interação. Nesses casos, os nomes dos objetos não são sublinhados.
- Foco de Controle: O foco de controle é um retângulo estreito que mostra o período durante o qual um objeto está executando uma ação. Ele pode ser

aninhado para indicar chamadas recursivas ou chamadas feitas por outros objetos. O foco de controle pode ser sombreado para mostrar a região em que o método do objeto está sendo executado.

- Mensagens: As mensagens são representadas por setas, indo de uma linha de vida para outra. Mensagens assíncronas têm setas finas, enquanto mensagens síncronas (chamadas) têm setas triangulares cheias. Uma resposta a uma mensagem síncrona é exibida por uma linha tracejada com uma seta fina. As mensagens de retorno podem ser omitidas, mas podem ser úteis para mostrar os valores de retorno.
- Ordenação Temporal e Sequenciamento: A ordenação temporal em uma única linha de vida é significante, enquanto a distância exata não é relevante. As linhas de vida não representam uma escala de tempo, e as posições das mensagens em linhas de vida separadas não implicam um sequenciamento de informações. No entanto, uma cadeia de mensagens estabelece uma cadeia de causalidade, onde um ponto em outra linha de vida no final da cadeia sempre segue o ponto na linha de vida original no início da cadeia.

Figura 21 – Representação de Diagrama de Sequência



Fonte: (UML 2 - Uma Abordagem Prática, 2018).

3 DESENVOLVIMENTO

Neste capítulo destacaremos imagens e diagramas do projeto. Nosso objetivo é proporcionar uma compreensão ampla do processo de criação e implementação.

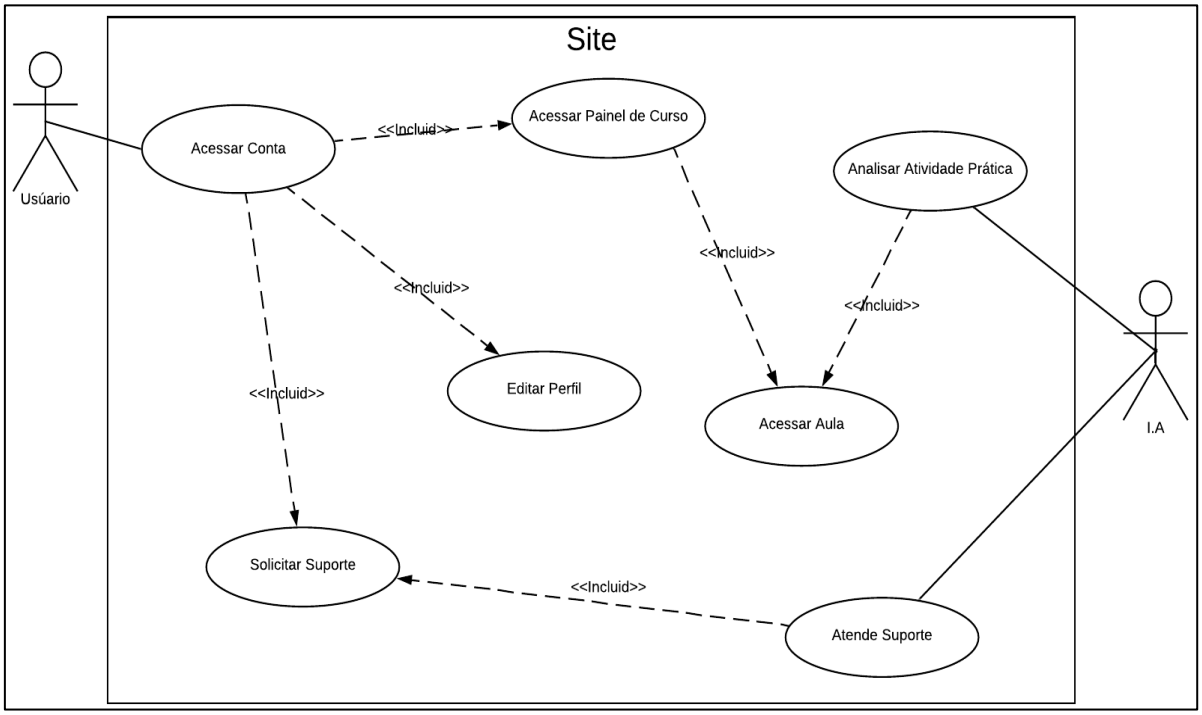
3.1 Diagramas do projeto

No contexto do desenvolvimento de projetos será apresentado os diagramas principais utilizados para a compreensão do desenvolvimento, a utilização de diagramas desempenha um papel fundamental na representação visual e na comunicação eficaz de conceitos, ideias e processos. Ao longo deste trabalho, abordaremos os diagramas de nosso projeto.

3.1.1 Caso de Uso

Aqui apresentamos nosso diagrama de caso de uso onde mostra de forma resumida como a dinâmica do usuário e a inteligência artificial irão se relacionar.

Figura 22 – Caso de uso Devschool

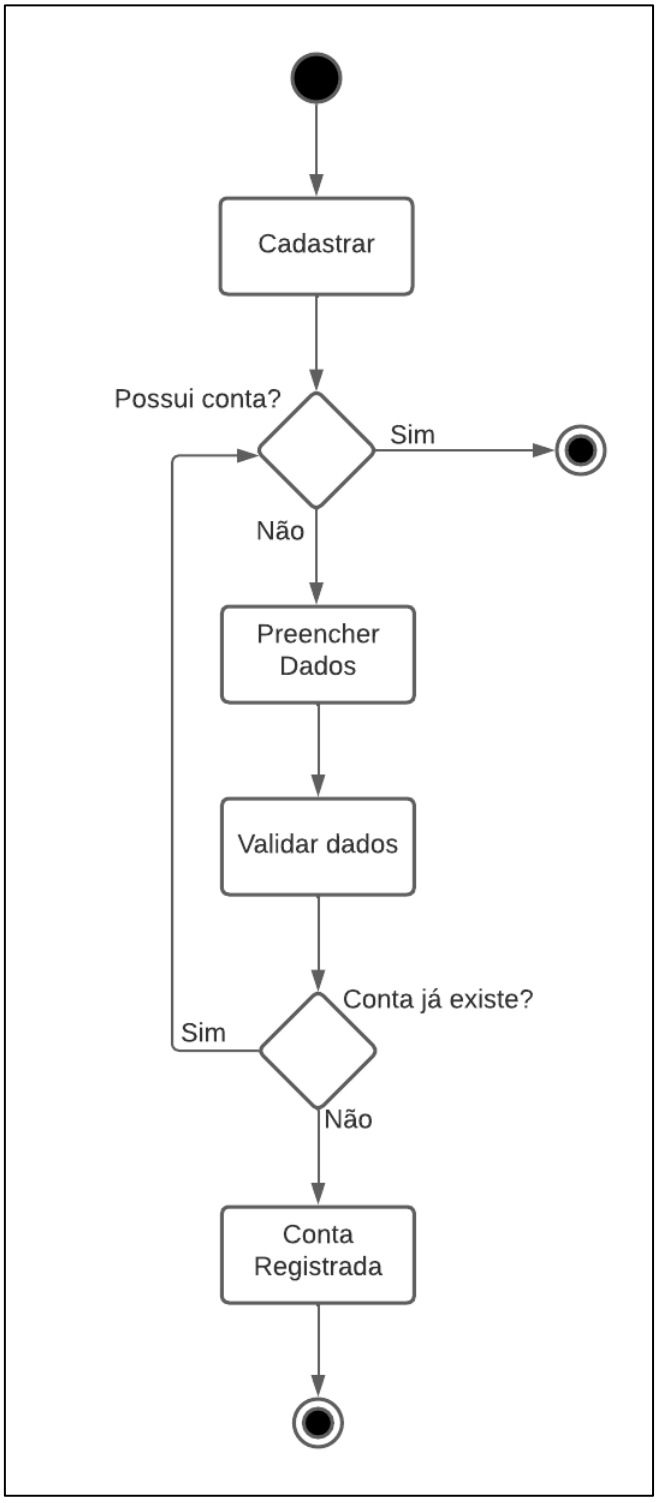


Fonte: Autoria própria, 2023.

3.1.2 Diagrama de Atividade

Apresentamos a seguir todos os diagramas de atividade, a seguir será mostrado as etapas do processo de cadastro no sistema:

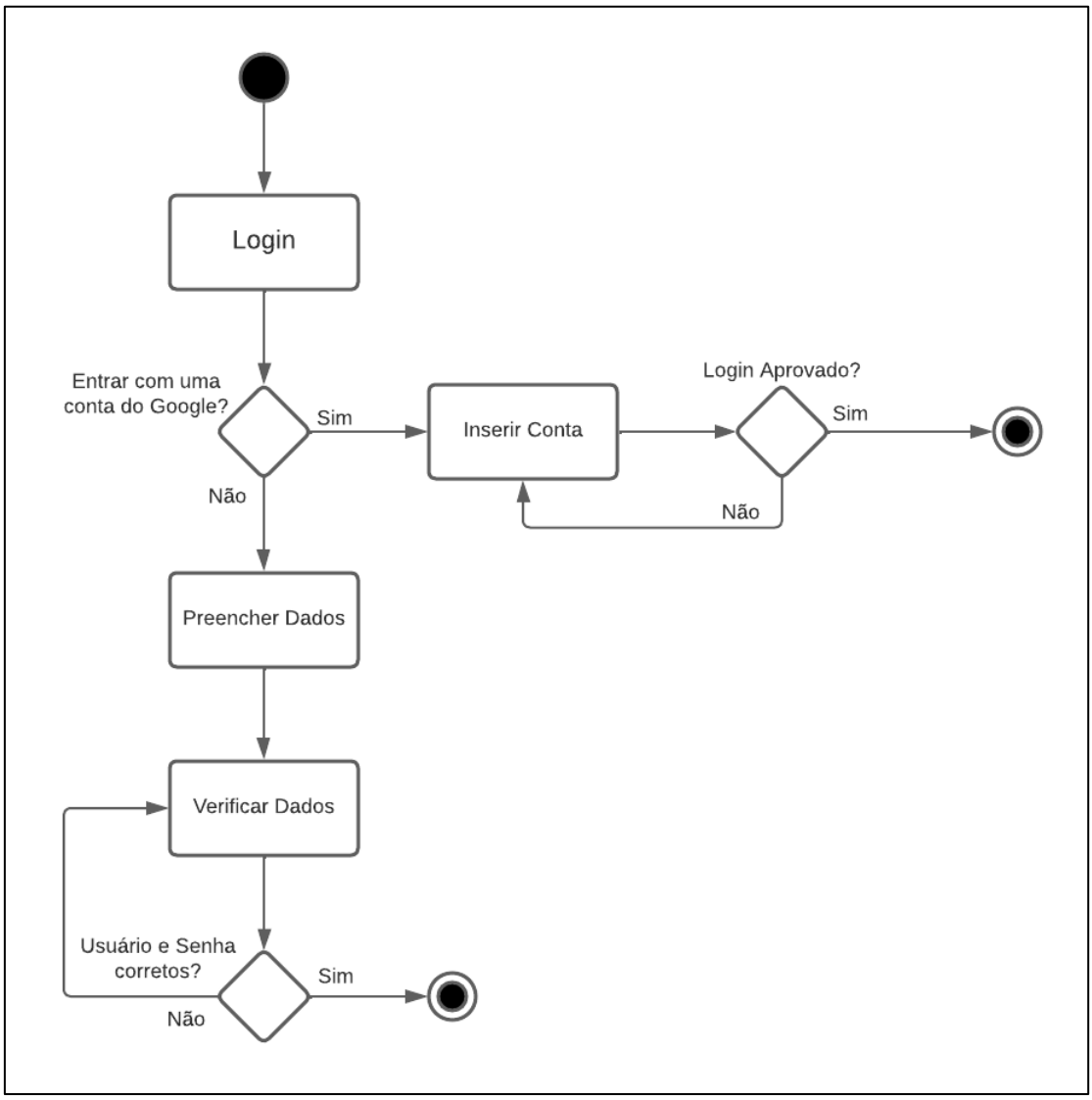
Figura 23 – Diagrama de atividade Cadastro



Fonte: Autoria própria, 2023.

A seguir o diagrama de atividade de *login*:

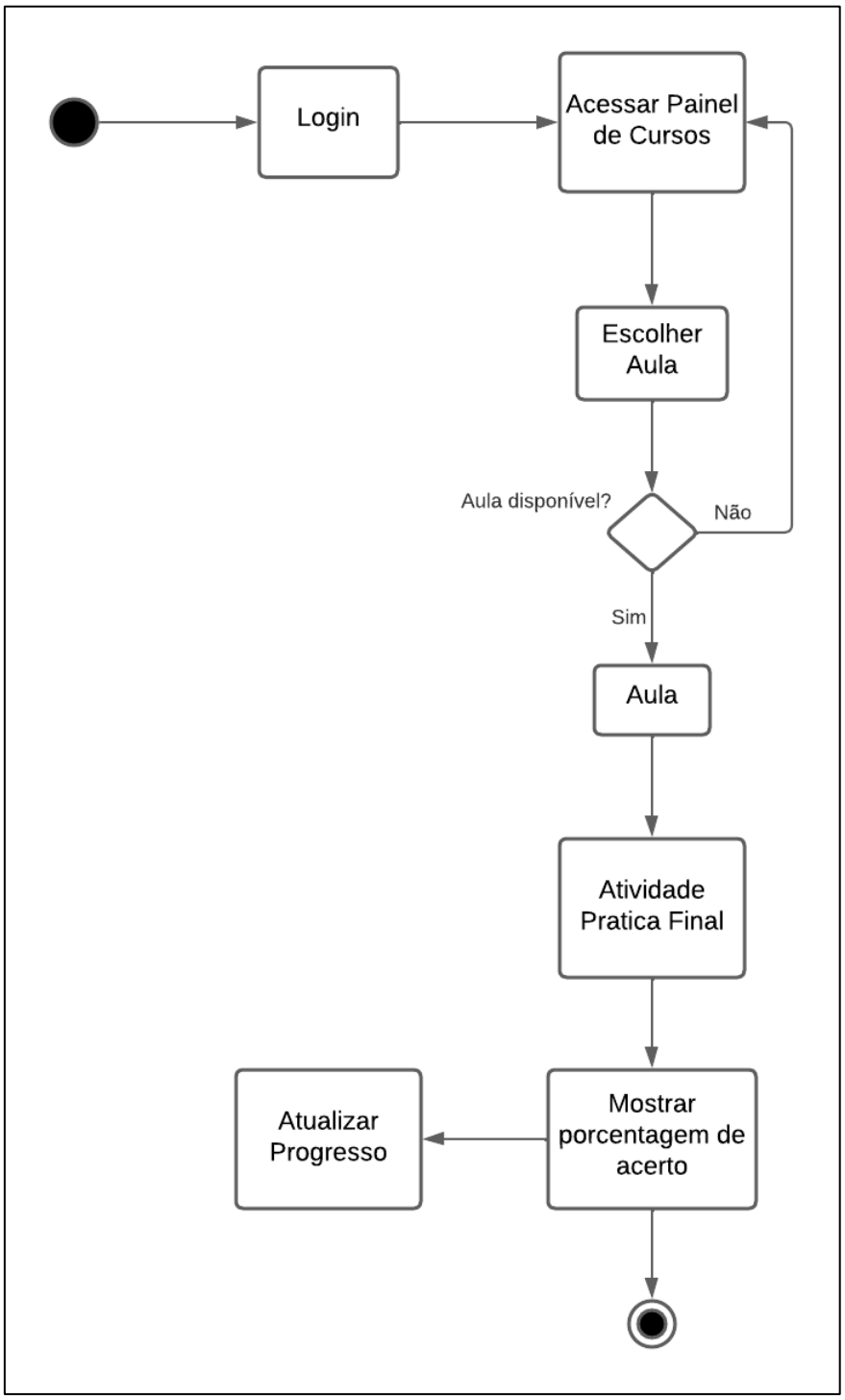
Figura 24 – Diagrama de atividade Login



Fonte: Autoria própria, 2023.

A seguir o diagrama de atividade das aulas:

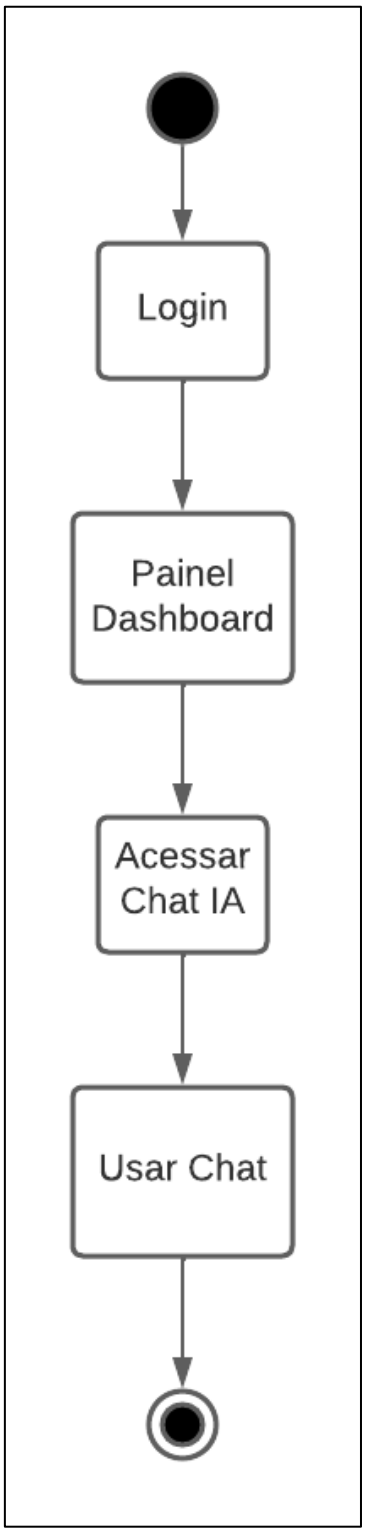
Figura 25 – Diagrama de atividade das aulas



Fonte: Autoria própria, 2023.

A seguir o diagrama de atividade do *Chat* IA:

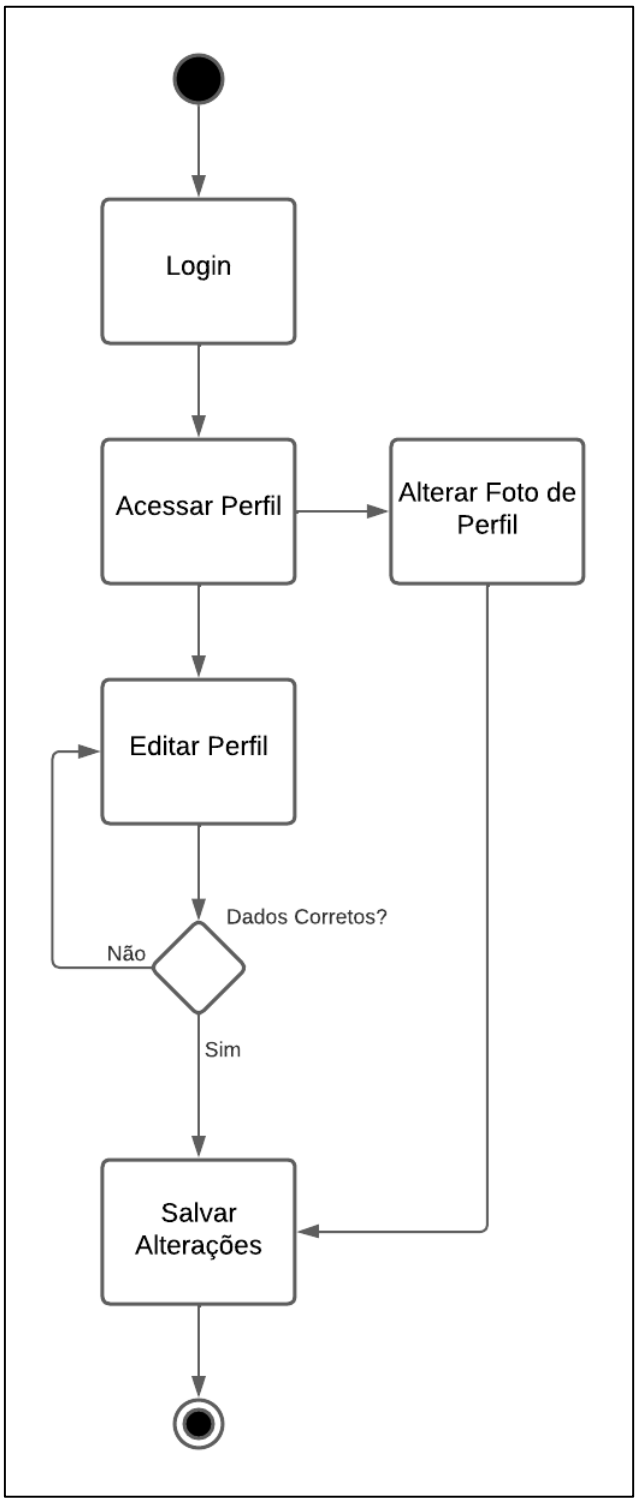
Figura 26 – Diagrama de atividade do Chat IA



Fonte: Autoria própria, 2023.

A seguir o diagrama de atividade de Perfil:

Figura 27 – Diagrama de atividade Perfil

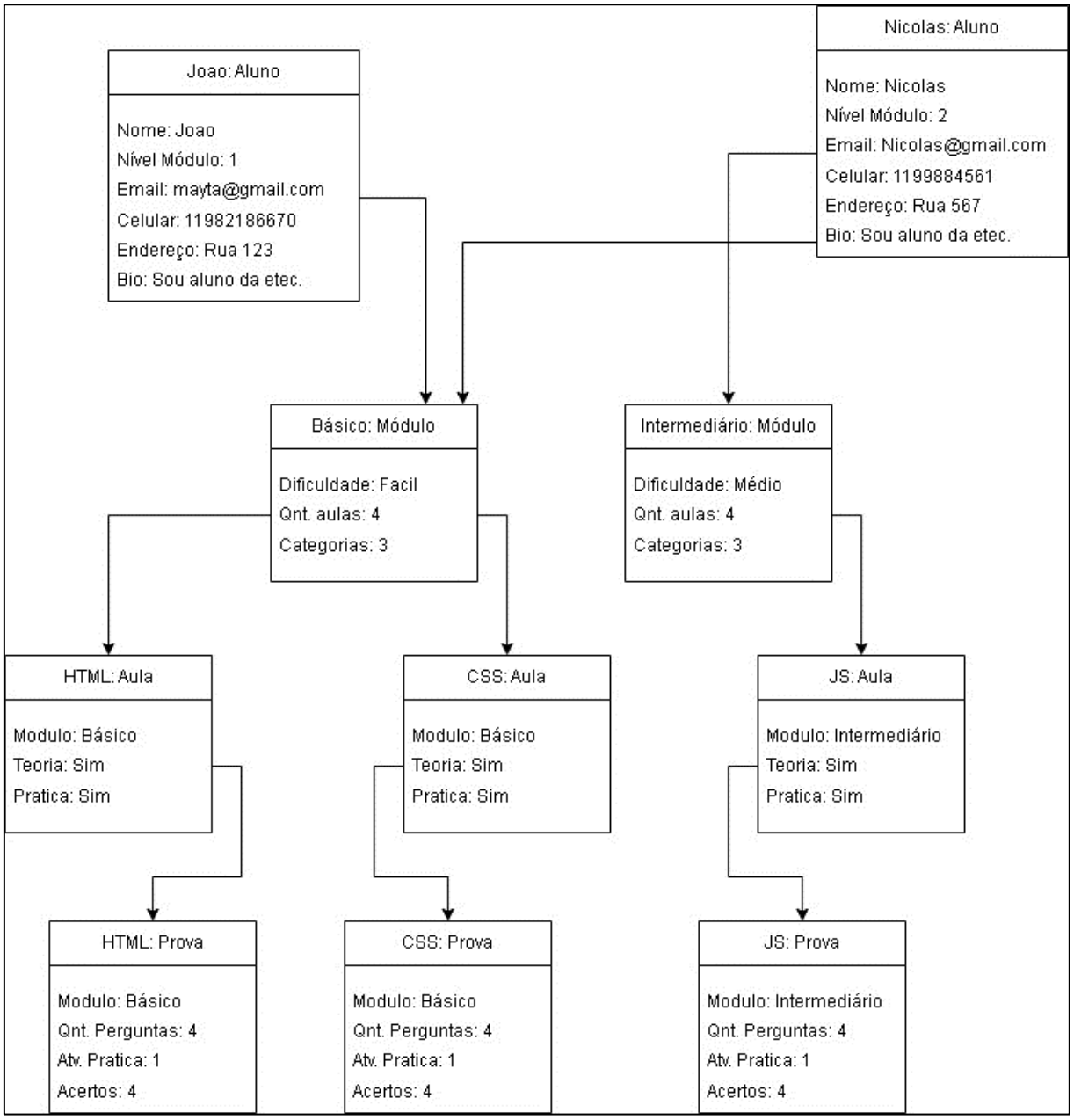


Fonte: Autoria própria, 2023.

3.1.3 Diagrama de Objetos

Logo abaixo apresentamos o nosso diagrama de objetos onde tem como objetivo mostrar a visão estática do sistema, onde ficarão guardadas as informações dos usuários e entre outras coisas:

Figura 28 – Diagrama de Objetos Devschool

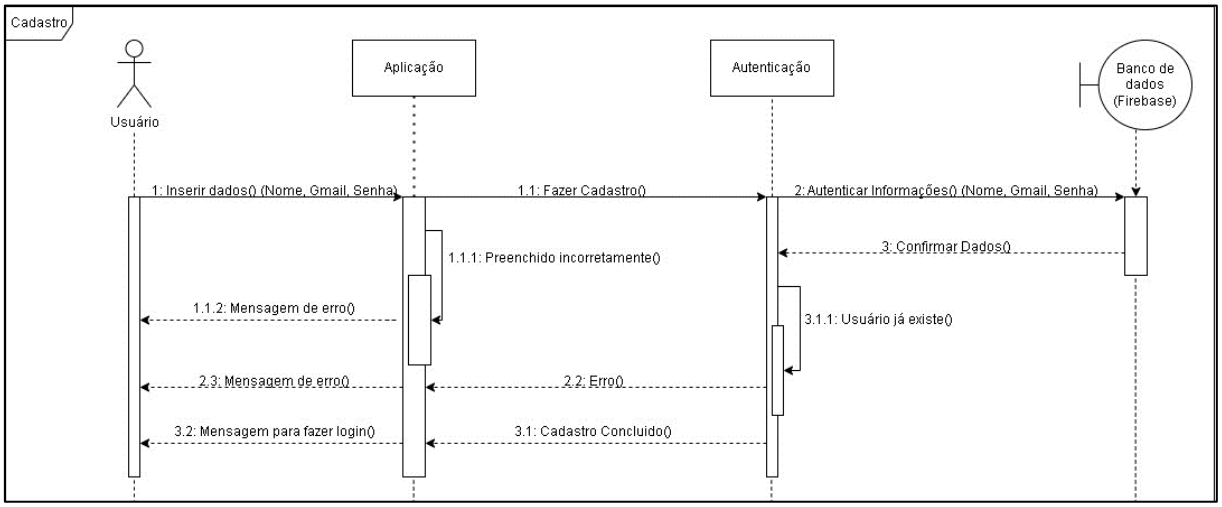


Fonte: Autoria própria, 2023.

3.1.4 Diagrama de Sequência

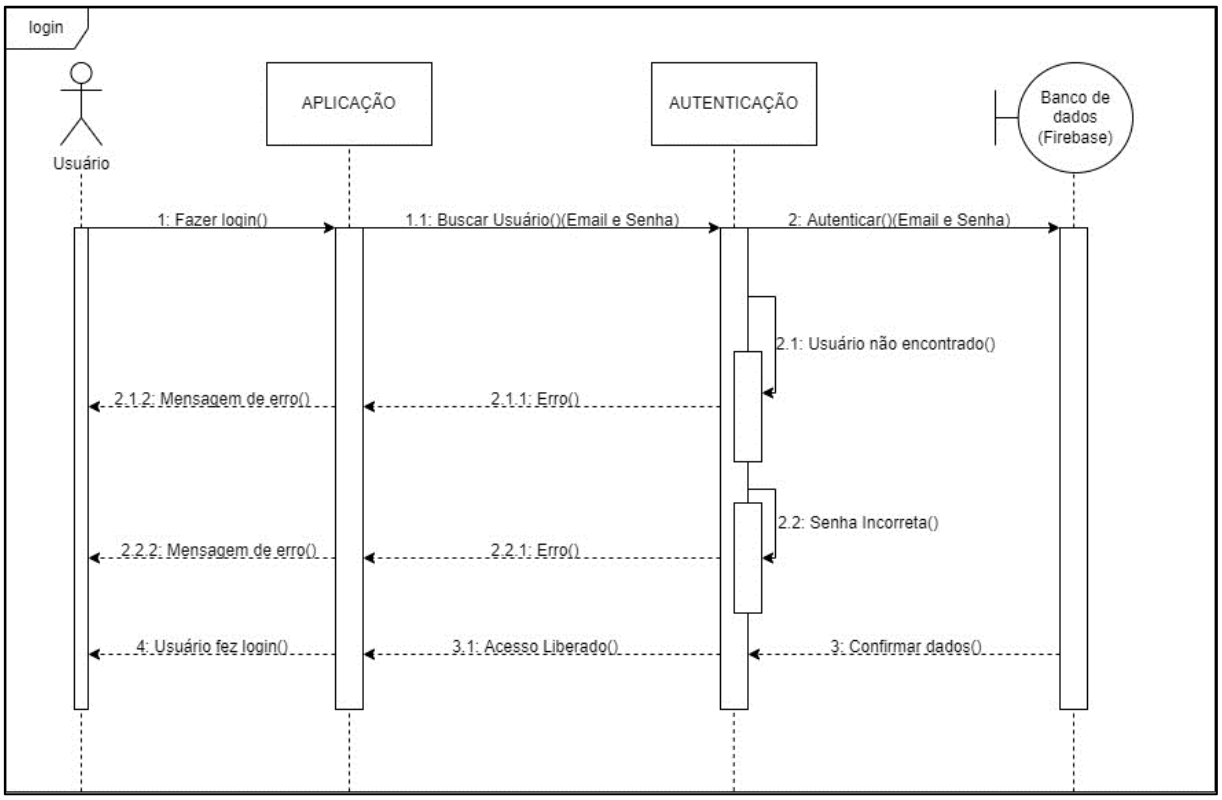
Logo abaixo apresentaremos duas figuras que simulam o cadastro e login de nosso sistema utilizando o diagrama de sequência:

Figura 29 – Cadastro diagrama de sequência



Fonte: Autoria própria, 2023.

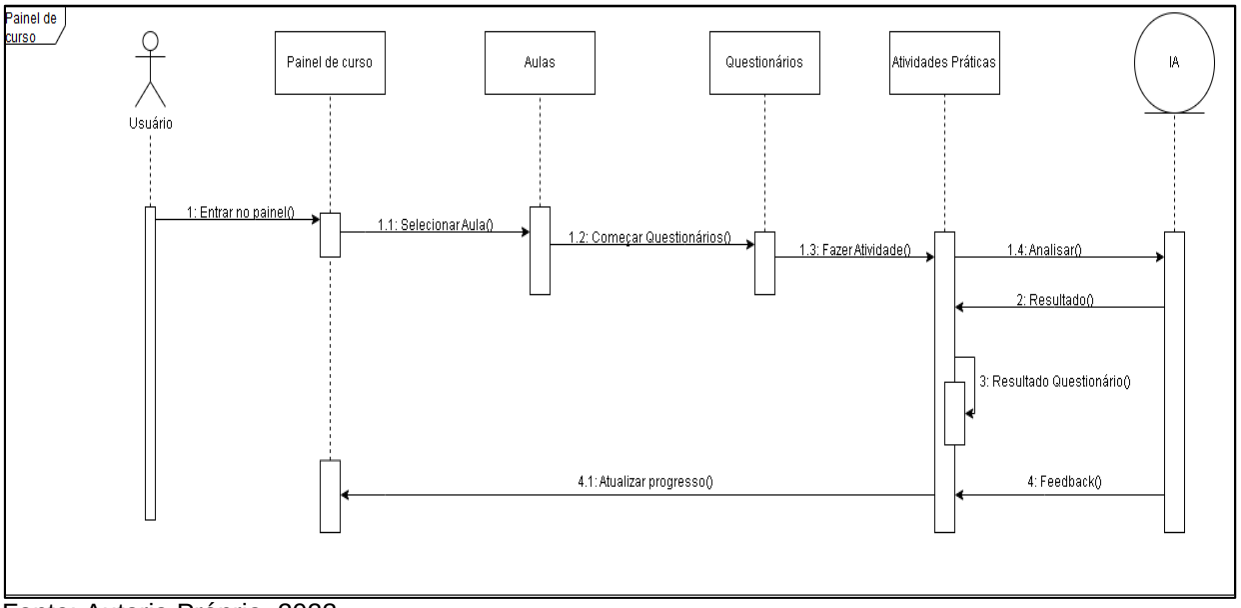
Figura 30 – Login diagrama de sequência



Fonte: Autoria Própria, 2023.

A seguir o diagrama de sequência de painel de curso:

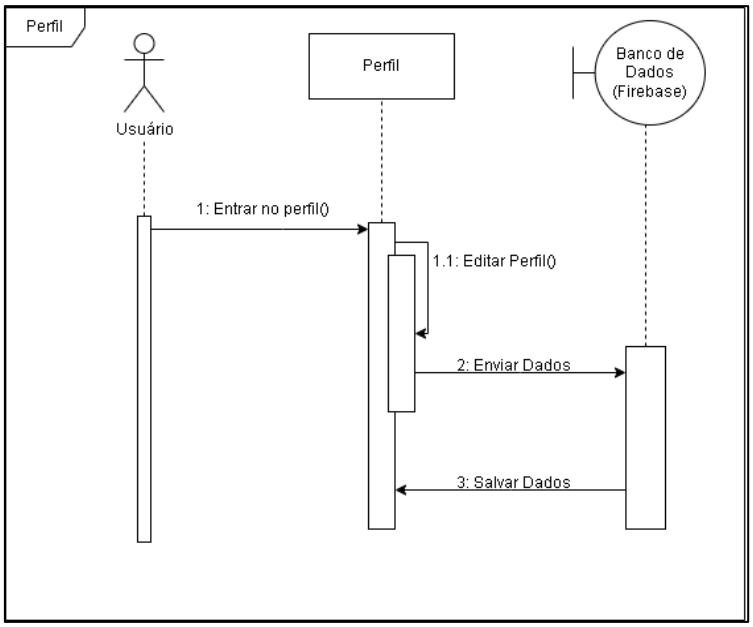
Figura 31 – Painel de curso diagrama de sequência



Fonte: Autoria Própria, 2023.

A seguir o diagrama de sequência de Perfil:

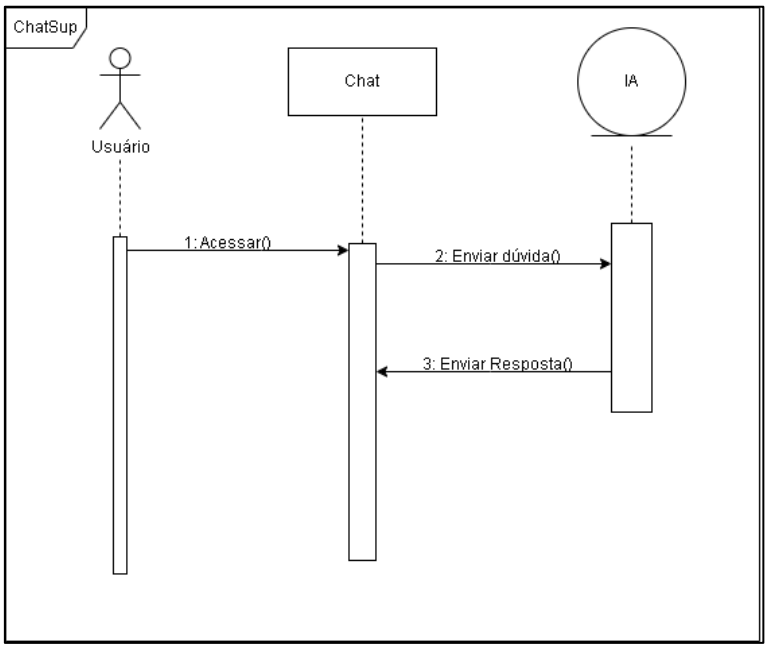
Figura 32 – Perfil diagrama de sequência



Fonte: Autoria Própria, 2023.

A seguir o diagrama de sequência de chat suporte:

Figura 33 – Chat suporte diagrama de sequência



Fonte: Autoria Própria, 2023.

3.2 Site

Este trabalho propõe o desenvolvimento de um site de ensino de programação voltado para iniciantes, com o objetivo de fornecer uma abordagem interativa e personalizada para o aprendizado de linguagens de programação. O site irá oferecer recursos educacionais, exercícios práticos, e um perfil amplo para o estudante acompanhar suas realizações e progresso, visando proporcionar uma experiência de aprendizado envolvente e eficaz.

3.2.1 Landing Page

Figura 34 – Página inicial do site

Fonte: Autoria própria, 2023.

Esta *landing page* foi desenvolvida com o propósito de atrair visitante. Trata-se de uma ferramenta poderosa no âmbito do *marketing* digital, pois possibilita que as empresas obtenham informações dos visitantes, gerem leads qualificados e incentivem ações específicas.

3.2.2 Tela de cadastro e login

O usuário que se interessar com curso, terá sua chamada a uma nova página onde terá as opções de cadastrar ou entrar com uma conta existente.

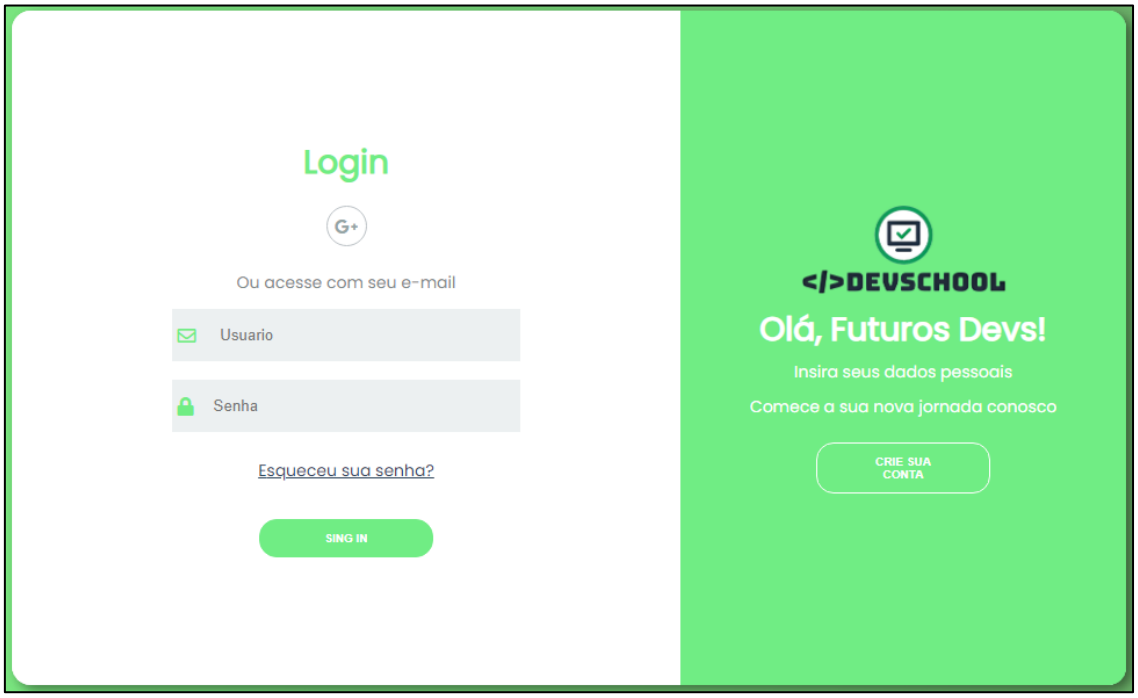
A tela de cadastro é essencial para permitir que os visitantes se tornem membros do site de ensino de programação. Ao criar uma conta preenchendo o usuário, e-mail e senha, o aluno terá acesso a recursos, como cursos, materiais de estudo e mais.

Figura 35 – Tela de cadastro do aluno

Fonte: Autoria própria, 2023.

A tela de *login* é projetada para permitir que os membros acessem facilmente sua conta e continuem de onde pararam, tendo a opções de entrar com uma conta Google. Além disso, a tela de *login* é importante para garantir a segurança dos seus dados e proteger suas informações pessoais.

Figura 36 – Tela de login do aluno



Fonte: Autoria própria, 2023.

3.2.3 Painel do aluno

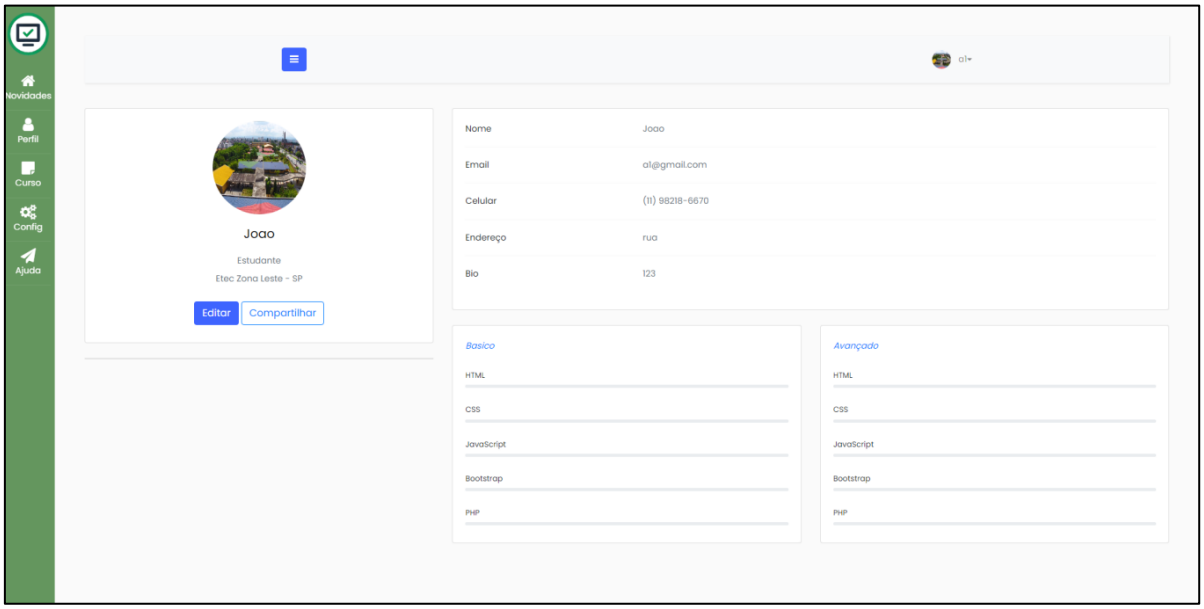
O painel do aluno é a parte que localiza as funções do curso, tendo como sua página inicial a tela "*Home*" da *dashboard* do aluno é o ponto central para se manter atualizado sobre as últimas novidades e informações relevantes relacionadas ao site de ensino de programação.

Figura 37 – Página inicial do painel do aluno

Fonte: Autoria própria (2023).

Na tela "Perfil", permite que o aluno visualize e atualize seus detalhes de contato, como nome, endereço de e-mail, foto de perfil e outras informações relevantes. Além disso, você pode fornecer uma breve descrição sobre si mesmo, destacar suas habilidades e experiência em programação, tornando o seu perfil mais completo e profissional.

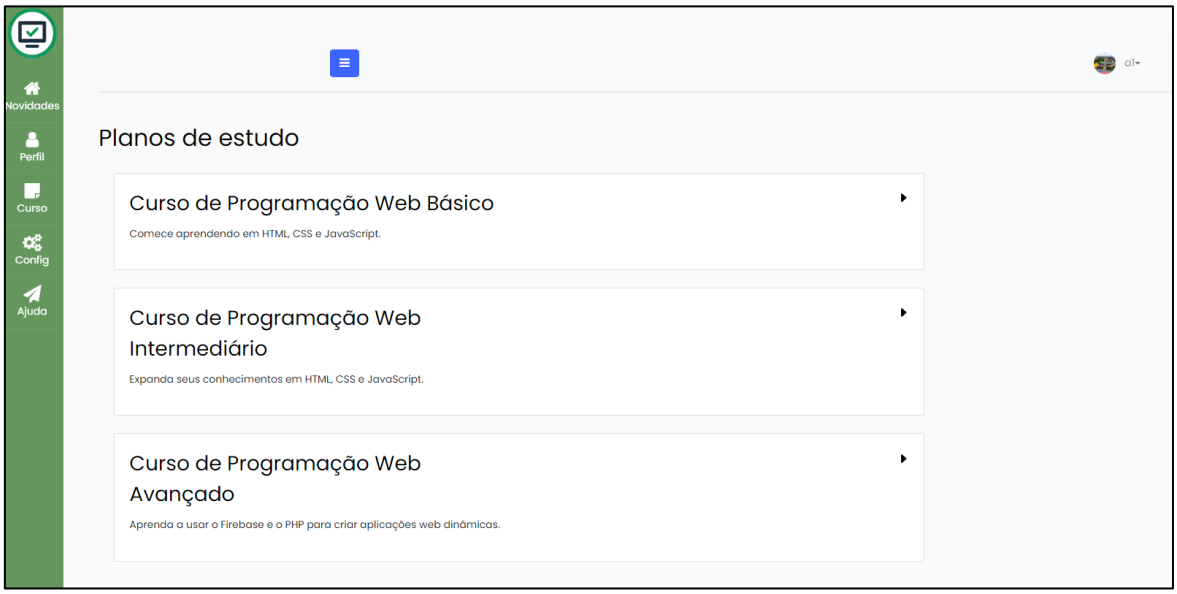
Figura 38 – Tela de perfil do aluno



Fonte: Autoria própria, 2023.

A tela "Curso" oferece acesso ao painel de módulos do curso e às aulas relacionadas. Aqui, o aluno pode visualizar todos os cursos em que está matriculado, navegar pelos módulos e tópicos específicos de cada curso e acessar as aulas correspondentes.

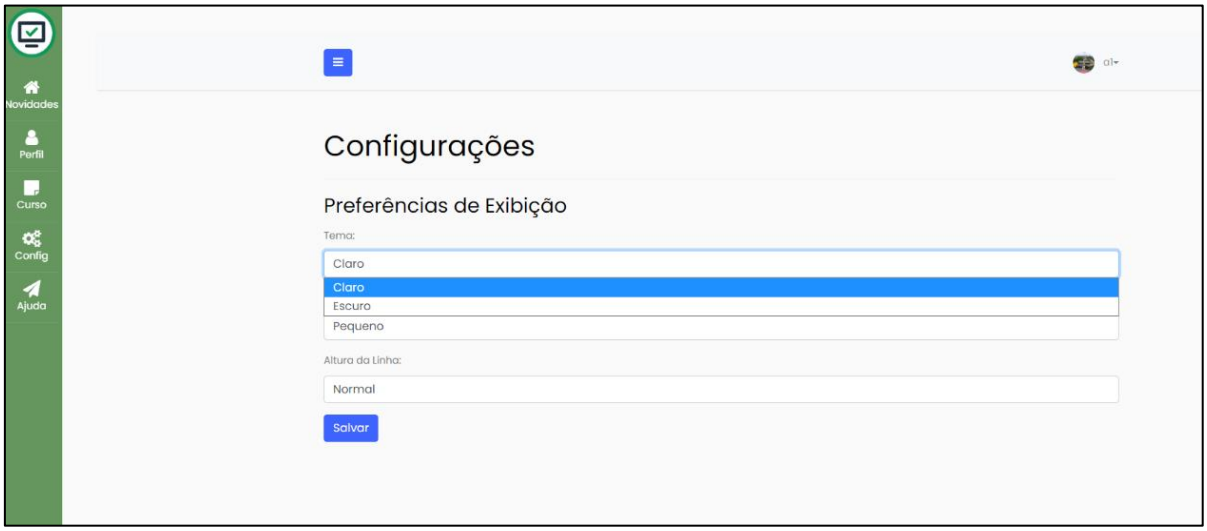
Figura 39 – Painel de aulas



Fonte: Autoria própria, 2023.

Na tela "Configurações", tem as opções para personalizar a experiência de uso da plataforma de ensino de programação. Aqui, o aluno pode ajustar as preferências de tela, como ativar o modo escuro para uma visualização mais confortável em ambientes de pouca luz, ajustar o tamanho do texto para uma leitura mais fácil e adaptar outras configurações de acordo com suas preferências pessoais.

Figura 40 – Tela de configurações



Fonte: Autoria própria, 2023.

Na tela "Ajuda", o usuário terá acesso a um *chat* integrado com inteligência artificial para obter suporte e esclarecer dúvidas. Essa funcionalidade permite perguntas ou problemas específicos relacionados aos cursos ou ao funcionamento da plataforma. A inteligência artificial responderá às suas perguntas e fornecerá orientações úteis para ajudá-lo a solucionar problemas técnicos, obter informações adicionais ou esclarecer dúvidas sobre o conteúdo do curso.

Figura 41 – Tela de ajuda

Fonte: Autoria própria, 2023.

3.2.4 Aulas

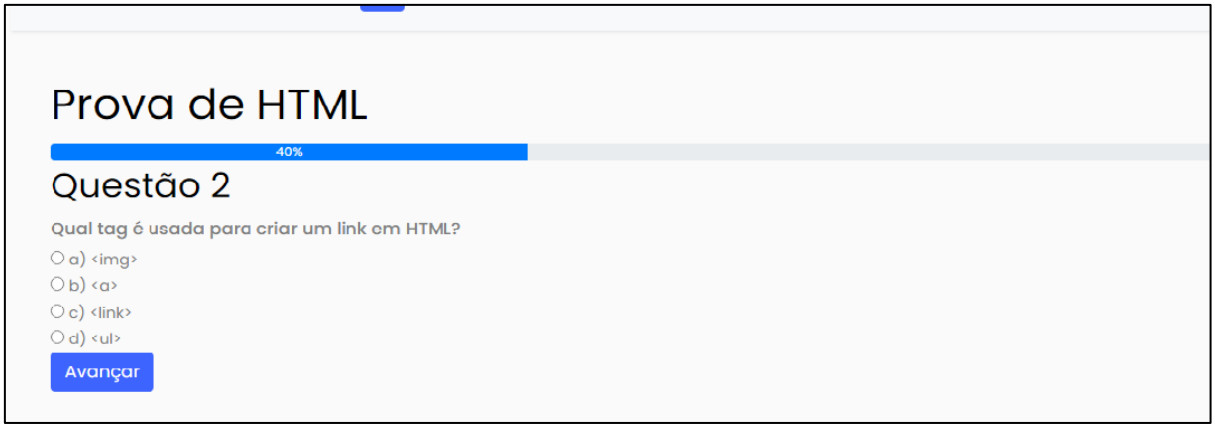
Ao usuário que acessar a aula, inicialmente precisara completar a aula teórica, contendo um conteúdo didático acerca da aula escolhida.

Figura 42 – Aula teórica de HTML

Fonte: Autoria própria, 2023.

Ao final da aula teórica o aluno prosseguira com o início da prova, o qual a quantidade de perguntas dependerá com a aula feita, no exemplo abaixo temos a prova de HTML básico contendo 5 questões.

Figura 43 – Prova de HTML



Fonte: Autoria própria, 2023.

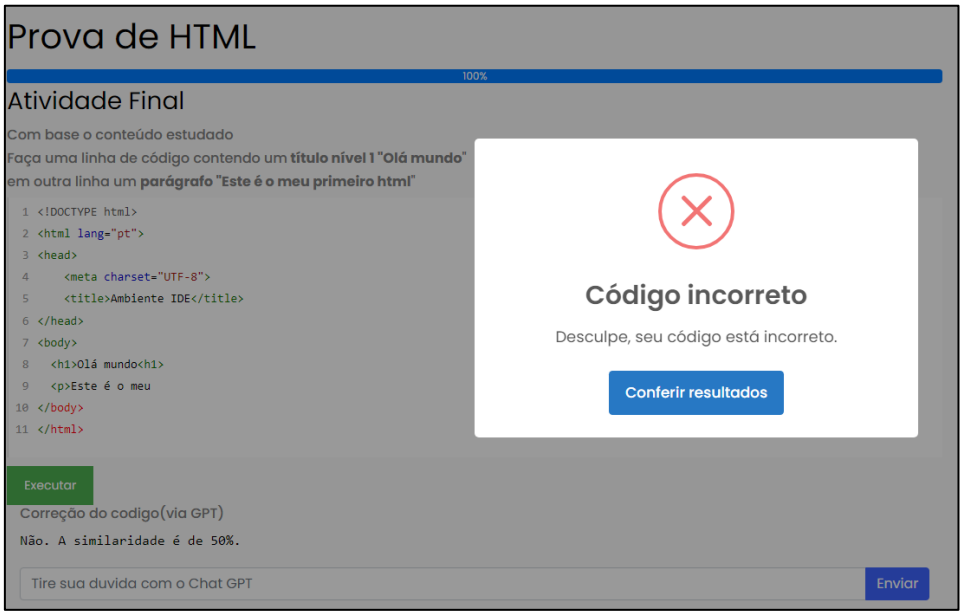
3.2.5 Atividade prática

Ao final de cada prova terá uma atividade prática, onde o aluno recebe instruções e deve aplicar os conceitos aprendidos no painel de texto, digitando o código, para que este seja analisado e corrigido pela inteligência artificial.

Figura 44 – Atividade prática de HTML

Fonte: Autoria própria, 2023.

Figura 45 – Correção feita pela inteligência artificial



Fonte: Autoria própria, 2023

4 CONSIDERAÇÕES FINAIS

Diante disso, o desenvolvimento do site proposto como metodologia de ensino à programação visa abordar e solucionar problemas relevantes, exclusivamente no ensino de programação. Ao oferecer um ambiente virtual acessível e interativo, aliado ao suporte de uma Inteligência Artificial (I.A), busca-se eliminar as barreiras enfrentadas por iniciantes e garantir uma jornada de aprendizagem abrangente, desde o básico até o avançado.

Durante o desenvolvimento do projeto, foram utilizadas linguagens de programação essenciais. O HTML estruturou o conteúdo, o CSS estilizou o visual e o JavaScript criou interatividade. Além disso, o PHP cuidou da lógica do servidor e da conexão com o banco de dados Firebase. Essas linguagens foram escolhidas por sua ampla adoção e recursos disponíveis. A combinação delas resultou em um ambiente virtual robusto, acessível e interativo para aprimorar a experiência de aprendizagem dos usuários.

Uma das principais contribuições desse projeto é a superação da falta de informações e orientações claras para aqueles que desejam iniciar no mundo da programação. Ao fornecer recursos educacionais estruturados e guias passo a passo, o site capacita os estudantes a dar os primeiros passos de forma consistente e encorajadora.

Além disso, o suporte de uma Inteligência Artificial permite que os alunos tenham acesso a um assistente virtual inteligente, capaz de oferecer respostas e soluções personalizadas para suas dúvidas e desafios. Essa interação baseada em I.A cria um ambiente de aprendizagem adaptativo, que se ajusta às necessidades e ao ritmo de cada aluno, promovendo uma experiência educacional mais eficaz e envolvente.

Outro ponto relevante é a contribuição para evitar defasagem e evasão de alunos em escolas de programação. Muitas vezes, estudantes chegam a essas instituições com lacunas de conhecimento ou dificuldades em conceitos básicos. O site proposto atua como uma ferramenta complementar, preenchendo essas lacunas e nivelando o conhecimento, preparando melhor os alunos para os desafios futuros.

É importante ressaltar que o projeto do site de ensino à programação não substitui o papel do professor ou a importância de uma formação presencial. Em vez disso,

busca-se fortalecer o processo de ensino-aprendizagem, fornecendo uma plataforma que amplia o acesso à informação, incentiva a prática e oferece suporte contínuo aos estudantes.

Considerando a crescente demanda por profissionais qualificados em programação e a importância do domínio dessa habilidade no cenário atual, o site projetado como metodologia de ensino à programação surge como uma solução promissora. Ao oferecer recursos abrangentes, interativos e personalizados, tem o potencial de empoderar estudantes e contribuir para o desenvolvimento de habilidades técnicas indispensáveis no mercado de trabalho.

REFERÊNCIAS

EIS, Diego; FERREIRA, Elcio. **HTML5 e CSS3**: com Farinha e Pimenta. 2012. 219p. Universidade Federal de Pernambuco (UFPE), Recife, Brasil. Disponível em: https://www.cin.ufpe.br/~dfop/Arquivos/Pacote%20Web/HTML5%20e%20CSS3%20com%20Farinha%20e%20Pimenta%20Diego%20Eis%20e%20Elcio%20Ferreira.pdf.

Acesso em: 19 de maio de 2023.

BORGES, Marcos Augusto F. **Avaliação de uma metodologia alternativa para a aprendizagem de programação**. 15 f. TCC (Graduação) - Curso de Programação, Faculdade Campo Limpo Paulista, São Paulo, 2000. 15p.

MOREIRA, Mireille Pinheiro. **Um ambiente para ensino de programação com feedback automático de exercícios**. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2017. 120 p.

CAELUM. **Estruturação de páginas usando HTML e CSS.** Editora: Alura, 2023, 296p. Disponível em: https://www.caelum.com.br/apostila/apostila-html-css-javascript.pdf. Acesso em: 13 de junho de 2023

DUCKETT, Jon. **HTML e CSS: Projete e construa websites**. 2. ed. São Paulo: Alta Books, 2016. 490 p.

SAMY, Maurício. **Bootstrap 3.3.5: Aprenda a usar o framework Bootstrap para criar layouts CSS complexos e responsivos.** São Paulo: Novatec Editora, 2015. 226p.

FLANAGAN, David. **JavaScript: O Guia Definitivo**. 6ª ed. São Paulo: Novatec Editora. 2013. 1096p.

PRESCOTT, Preston. **Programação em JavaScript**. Casa do Código, 2016. 228 p.

MORRISON, Michael. **Use a Cabeça! Javascript**. 1st ed. Rio de Janeiro: Alta Books, 2008. 462p.

DALL'OGLIO, Pablo. **PHP Programando com Orientação a Objetos**. 3ª ed. São Paulo: Novatec Editora, 2016. 576p.

CONVERSE, Tim; PARK, Joyce. **PHP: a bíblia**. São Paulo: Pearson Education, 2002. 855p.

GOOGLE. **Firebase**. Disponível em: https://firebase.google.com/docs?hl=pt. Acesso em: 11 mai. 2023.

RUSSEL Stuart; NORVIG Peter. **Inteligência Artificial**. Rio de janeiro: GEN LTC, 2013. 1016p.

GOMES, Dennis dos Santos. **Inteligência Artificial**: Conceitos e Aplicações. Rondônia: Acadêmico das Faculdades Associadas de Ariquemes- FAAr, 2010. 246p.

OPENAI. **ChatGPT**. Disponível em: platform.openai.com/docs/introduction/overview. Acesso em: 11 mai. 2023.

BOOCH, Grady. **Uml - Guia do Usuário**. 2. ed. Rio de Janeiro: Elsevier, 2012. 552p.

GUEDES, Gilleanes T. **UML 2 - Uma Abordagem Prática**: 3ª ed. 2018. 352p. São Paulo: Novatec Editora.

PERA, Bruno. (Ano). **Apostila de Banco de Dados V3**. 37p. Disponível em: https://www1.univap.br/bruno.pera/uploads/INFORMATICA/BANCODEDADOS/Apostila_de_Banco_de_Dados_V3.pdf . Acessado em: 06 junho. 2023.

ALVES, William Pereira. **Banco de dados: Teoria e Desenvolvimento.** 2020. 590p. São Paulo: Érica.