

CENTRO ESTADUAL DE EDUCAÇÃO PAULA SOUZA
ESCOLA TÉCNICA ESTADUAL ZONA LESTE
Mtec Desenvolvimento De Sistemas AMS

João Pedro Bispo de Biasi

Lucas Bonfim Vilela

Neluma Siqueira Lopes

BLOOMY: Sistema Interativo Para Amparo Educacional.

São Paulo

2025

João Pedro Bispo de Biasi

Lucas Bonfim Vilela

Neluma Siqueira Lopes

**BLOOMY: Sistema Interativo No Amparo Educacional Do Ensino
Básico.**

Trabalho de Conclusão de Curso apresentado ao
Mtec Desenvolvimento de Sistemas AMS da Etec
Zona Leste, orientado pelo Prof. Esp. Jeferson
Roberto de Lima, como requisito parcial para a
obtenção do título de técnico em Desenvolvimento de
Sistemas.

São Paulo

2025

BLOOMY

Sistema Interativo No Amparo Educacional Do Ensino Básico.

João Pedro Bispo de Biasi

Lucas Bonfim Vilela

Neluma Siqueira Lopes

Aprovada em __/__/____.

BANCA EXAMINADORA

Prof. Esp. Jeferson Roberto de Lima

Universidade do Jeferson

Prof. (Professor avaliador)

Universidade do Avaliador

Prof. (Professor avaliador)

Universidade do Avaliador

Página de Dedicatória

Este trabalho é dedicado a todos que por algum motivo sentiram que a escola não era o seu lugar, e sentem a necessidade de desistir. Porque, esse não é um desejo repentino, é acumulativo. Mesmo que não acredite que é o suficiente para exercer alguma função, por conta de não ser muito bom em alguma matéria desde o Fundamental.

Página de Agradecimientos

“O começo é sempre
hoje.”

**Mary Wollstonecraft
Shelley**

RESUMO

A princípio, diversos fatores podem contribuir para o baixo rendimento escolar. Entre eles, a insuficiência de investimentos para a infraestrutura de instituições de ensino e a fragilidade no modelo didático utilizado na atualidade. Contudo, a pesquisa realizada tem como base o desinteresse expresso por parte dos alunos, sobretudo as crianças do Fundamental I na Zona Leste de São Paulo. Conforme esse grupo é progressivamente inserido no ambiente digital, tornam-se notáveis os novos desafios para a educação, o que exige um método eficaz que desperte o interesse e melhore o desempenho dos alunos nos anos iniciais. O propósito dessa pesquisa é uma aplicação multiplataforma de jogos sérios acompanhada de um controle com estímulos sensoriais, promovendo a atenção e engajamento dos alunos, o que contribui para progresso da sua jornada escolar.

Palavras-Chave: Rendimento escolar; desinteresse estudantil; jogos sérios; aplicação multiplataforma; Ensino Fundamental I.

ABSTRACT

Several factors can contribute to academic underperformance, including insufficient investments in the infrastructure of educational institutions and the weaknesses in the current teaching model. This research focuses on the lack of interest shown by students, especially children in Elementary School I in the Eastern Zone of São Paulo, Brazil. As these students are progressively introduced to digital environments, new challenges in education become apparent, which require an effective method that awakens interest and improves the performance of students in their early years. The purpose of this study is to develop multiplatform serious games application, paired with a control with sensory stimuli, promoting attention and engagement of students. Which contributes to the progress and performance throughout their school journey.

Keywords: Academic performance; student disinterest; serious games; multiplatform application; Elementary School I.

LISTA DE ILUSTRAÇÕES

<i>Figura 01 — Exemplo Diagrama de Caso de Uso.....</i>	<i>20</i>
<i>Figura 02 — Exemplo Diagrama de Atividade.....</i>	<i>23</i>
<i>Figura 03 — Exemplo Diagrama de Sequência</i>	<i>24</i>
<i>Figura 04 — Exemplo Diagrama de Máquina de Estados.....</i>	<i>25</i>
<i>Figura 05 — Exemplo Wireframe de Baixa Fidelidade.....</i>	<i>27</i>
<i>Figura 06 — Exemplo Wireframe de Alta Fidelidade.....</i>	<i>28</i>
<i>Figura 07 — Exemplo Código em HTML.....</i>	<i>30</i>
<i>Figura 08 — Exemplo Formulário em HTML.....</i>	<i>31</i>
<i>Figura 9 — Exemplo Código em CSS</i>	<i>32</i>
<i>Figura 10 — Exemplo Código em HTML e CSS</i>	<i>33</i>
<i>Figura 11 — Exemplo Formulário em CSS.....</i>	<i>34</i>
<i>Figura 12 — Exemplo Código em JS</i>	<i>35</i>
<i>Figura 13 — Exemplo Código em JS e HTML.....</i>	<i>35</i>
<i>Figura 14 — Exemplo Formulário em JS.....</i>	<i>36</i>
<i>Figura 15 — Exemplo código em React.....</i>	<i>37</i>
<i>Figura 16 — Exemplo Formulário em React</i>	<i>38</i>
<i>Figura 17 — Exemplo de Código em React Native</i>	<i>39</i>
<i>Figura 18 — Exemplo de Código em React Native</i>	<i>39</i>
<i>Figura 19 — Exemplo de Código em React Native</i>	<i>40</i>
<i>Figura 20 — Exemplo de Código em React Native</i>	<i>40</i>
<i>Figura 21 — Exemplo de Código em React Native</i>	<i>41</i>
<i>Figura 22 — Exemplo de Código em React Native</i>	<i>42</i>
<i>Figura 23 — Exemplo Formulário em React Native</i>	<i>43</i>
<i>Figura 24 — Exemplo Código em TS.....</i>	<i>44</i>
<i>Figura 25 — Exemplo de código em C++.....</i>	<i>47</i>
<i>Figura 26 — Microcontrolador ESP8266.....</i>	<i>49</i>
<i>Figura 27 — Wireframe de Baixa Fidelidade 01</i>	<i>56</i>
<i>Figura 28 — Wireframe de Alta Fidelidade 02.....</i>	<i>57</i>
<i>Figura 29 — Wireframe de Baixa Fidelidade 02.....</i>	<i>57</i>
<i>Figura 30 — Wireframe de Alta Fidelidade 02.....</i>	<i>58</i>
<i>Figura 31 — Wireframe de Baixa Fidelidade 03.....</i>	<i>59</i>
<i>Figura 32 — Wireframe de Alta Fidelidade 03.....</i>	<i>60</i>
<i>Figura 33 — Wireframe de Baixa Fidelidade 04.....</i>	<i>61</i>

<i>Figura 34 — Wireframe de Alta Fidelidade 04.....</i>	<i>62</i>
<i>Figura 35 — Wireframe de Baixa Fidelidade 05.....</i>	<i>63</i>
<i>Figura 36 — Wireframe de Alta Fidelidade 05.....</i>	<i>64</i>
<i>Figura 37 — Wireframe de Baixa Fidelidade 06.....</i>	<i>65</i>
<i>Figura 38 — Wireframe de Alta Fidelidade 06.....</i>	<i>66</i>
<i>Figura 39 — Wireframe de Baixa Fidelidade 07.....</i>	<i>67</i>
<i>Figura 40 — Wireframe de Alta Fidelidade 07.....</i>	<i>68</i>
<i>Figura 41 — Wireframe de Baixa Fidelidade 08.....</i>	<i>69</i>
<i>Figura 42 — Wireframe de Alta Fidelidade 08.....</i>	<i>70</i>
<i>Figura 43 — Wireframe de Baixa Fidelidade 09.....</i>	<i>71</i>
<i>Figura 44 — Wireframe de Alta Fidelidade 09.....</i>	<i>72</i>
<i>Figura 45 — Wireframe de Baixa Fidelidade 10.....</i>	<i>73</i>
<i>Figura 46 — Wireframe de Alta Fidelidade 10.....</i>	<i>74</i>
<i>Figura 47 — Sprites dos personagens.....</i>	<i>75</i>

LISTA DE ABREVIATURAS E SIGLAS

Application Programming Interface (API)

Unified Modeling Language (UML)

HyperText Markup Language (HTML)

Cascading Style Sheets (CSS)

JavaScript (JS)

TypeScript (TS)

Two-Dimensional (2D)

Internet of Things (IoT)

Bluetooth Low Energy (BLE)

Light Emitting Diode (LED)

Central Processing Unit (CPU)

Three-Dimensional (3D)

LISTA DE TABELAS

Tabela 1 — Exemplo da Documentação do Caso de Uso	21
---	----

SUMÁRIO

1 INTRODUÇÃO	15
2 REFERENCIAL TEÓRICO.....	17
2.1 Desamparo Educacional Sentido Nos Alunos do Ensino Básico	17
2.1.1 Jogos sérios	17
2.2 Tecnologias utilizadas	18
2.2.1 Design Thinking	18
2.2.2 Unified Modeling Language (UML)	18
2.2.2.1 Diagrama de Casos de Uso	19
2.2.2.2 Documentação dos Casos de Uso.....	20
2.2.2.5 Diagrama de Atividade	23
2.2.2.3 Diagrama de Sequência	24
2.2.2.4 Diagrama de Máquina de Estados	25
2.2.3 Wireframes	26
2.2.4 HTML	28
2.2.5 CSS	31
2.2.6 JavaScript	34
2.2.7 React	36
2.2.8 React Native.....	38
2.2.9 TypeScript	43
2.2.10 Banco de dados.....	44
2.2.11 Firebase.....	45
2.2.12 Construct	45
2.2.13 C++	46
2.2.14 Internet of Things (IOT).....	48
2.2.15 ESP32	48
2.2.16 Modelagem 3D	49

2.2.17 Impressão 3D.....	49
3 DESENVOLVIMENTO.....	51
3.1 Conceitos.....	51
3.2 Levantamento de Requisitos.....	51
3.2.1 Requisitos Funcionais	51
3.2.2 Requisitos Não Funcionais	52
3.2.3 Regras de Negócios	53
3.3 Modelagem e Documentação do Sistema	54
3.3.1 Diagrama de Casos de Uso	54
3.3.2 Documentação do Caso de Uso.....	54
3.3.3 Diagrama de Atividades	54
3.3.4 Diagrama de Sequência.....	55
3.3.5 Diagrama de Máquina de Estados	55
3.4 Estrutura da Plataforma Digital.....	55
3.5 Wireframes de Baixa e Alta Fidelidade.....	56
3.6 Desenvolvimento dos Jogos.....	74
3.6.1 Mecânicas Aplicadas	75
3.6.2 Conceitos Visuais.....	76
3.7 Construção do Controle Interativo	76
3.7.1 Circuito e Funcionamento	76
3.7.2 Modelagem e Impressão.....	76
4 CONSIDERAÇÕES FINAIS	77
REFERÊNCIAS.....	77

1 INTRODUÇÃO

Bloomy é um sistema interativo entre um controle sensorial e jogos sérios, onde utilizaremos de ferramentas digitais e embarcadas para visar a eficiência em amparar o sistema educacional básico.

Desenvolver um sistema para auxílio escolar é o que temos objetivado, colocando em foco ferramentas lúdicas como jogos e brincadeiras — a fim de cultivar maior interesse vindo dos alunos — e um controle sensorial — que por meio de reações específicas, almeja frisar situações dos jogos, fomentando o exercício de cada matéria escolhida.

Para isso, é necessário investigar e analisar as dificuldades dos alunos do primeiro fundamental, e tal como devemos averiguar as metodologias mais eficientes e flexíveis para incorpora-las na inovação; perante ao desenvolvimento, será acurado a confecção de interfaces amigáveis, intuitivas e confortáveis aos olhos; também, construir um hardware que comunique de forma eficiente com o software; contudo, programar jogos-sérios que ainda sejam interessantes para os alunos, estando em sincronia com o controle.

É notável o descaso com o aprendizado vindo pelos estudantes, que pontuam a escola como um ambiente infrutífero e desgastante para tal atividade — fato esse prejudicial para o futuro desses alunos (MARQUES; CASTANHO, 2011). Portanto, para equilibrar a discussão, abordagens inovadoras que sejam incisivas deveriam ser revisitadas por serem atuais e promissoras, como jogos epistêmicos (SENA, et. al, 2016).

Tal problema ocorre por diversos motivos — levando em conta a diversidade individual do aluno dentro e fora da escola. A incapacidade de suprir as dores cotidianas, é um dos fatores da evasão escolar, não necessariamente do ambiente, mas do conteúdo e do aprendizado.

A evasão escolar no Brasil, tem se agravado com cerca de 400 mil jovens de 6 a 14 anos deixando de frequentar a escola no ano de 2023, como analisado por Camila da Silva (2024). Este fato, é uma preocupação latente dos educadores e um foco de urgência nacional. Com isso, este projeto tem como objetivo atender a demanda por meios que ajudem os professores a estimularem o interesse dos alunos, e auxiliar na conclusão da base educacional, especialmente dos alunos com maior aptidão

sensorial. Embora existam outros projetos que utilizam de jogos e brincadeiras para melhorar o ensino, muitos não atendem as demandas necessárias e usufruem dos modelos engessados que se assemelham aos mais tradicionais de ensino, perpetuando a falta de interesse nas crianças. Por isso, a criação de uma plataforma de jogos educativos acompanhada de controles táteis e sensoriais se torna uma opção para a diminuição dos casos em que um aluno com maior aptidão sensorial saía da escola por falta de adequação ao ensino tradicional.

Para pesquisas e soluções em campos científicos e relacionados a educação, é importante entender o “como” e “porque” do objeto, aprofundando-se por inteiro no ambiente, grupo ou problemática ao recorrer da metodologia qualitativa (ATENA EDITORA, 2023, p. 13). Além disso, destacam-se o *Design Thinking* e as ferramentas da UML, para a consolidação do protótipo em relação às pesquisas realizadas; também utilizando o ESP32 com os conceitos de IOT e funções de BLE, em sincronia das aplicações baseadas em *React Native* que porta os jogos em *Construct 3*.

Este projeto visa amparar os alunos da Zona Leste de São Paulo, que estão matriculados no Fundamental 1, especialmente aqueles de escolas municipais onde o ensino é significativamente inferior comparado com escolas estaduais ou privadas.

Para falar sobre educação e seus desafios utilizamos Pedagogia da Autonomia de Paulo Freire; embasando os conceitos de React Native por Bruna Esculadeiro, e métodos de diagramação do Gilleanes Guedes.

No presente documento estão descritos a base teórica, o desenvolvimento, conclusão e referências utilizadas para a realização do projeto Bloomy.

2 REFERENCIAL TEÓRICO

Este capítulo discorre sobre toda a fundamentação teórica pesquisada para basear a construção do projeto.

2.1 Desamparo Educacional Sentido Nos Alunos do Ensino Básico

Um fato sobre o baixo desempenho acadêmico, discute a relação da má vontade de absorver os saberes passados pelo educador, que não ampara as vivências únicas do aluno, dificultando a trilha para uma educação libertadora (FREIRE, 1996). Pesquisas apontam a falta de investimentos na infraestrutura escolar brasileira, com instituições abaixo do padrão mínimo de qualidade: sem bibliotecas, livros e até mesmo diretoria — destacando o sucateamento do ensino (VASCÓNCELOS *et. al*, 2021). Além disso, sabe-se que a escola é o meio essencial para um futuro de qualidade, mas pouca importância é dada para o aprendizado; muitos estudantes julgam a escola como insatisfatória, sem sentido, cansativa e conflituosa (MARQUES; CASTANHO, 2011).

Com isso, de acordo com Samara de Sena *et. al.*, as mudanças do século XXI tornam evidente que a abordagem educacional deve ser revista e jogos epistêmicos é uma aposta promissora (SENA, *et. al*, 2016).

2.1.1 Jogos sérios

Como dito por Vasconcelos *et. al* (2017), estudiosos buscaram formas de vincular o brincar dos jogos com o aprendizado, chamados "jogos sérios", onde planejam usar de ferramentas lúdicas promovendo habilidades autônomas aos alunos. Segundo Paula (2015), jogos azem um mundo imersivo — sendo trilha para o subconsciente das pessoas — e qualquer mensagem, para ele é passível de reflexões intimistas pelo jogador, criando e desenvolvendo o pensamento crítico.

Porém, como visto atualmente, a incidência do uso excessivo de aparelhos eletrônicos cresceu, o que leva todo o foco e interesse por tarefas que te desafiam a diminuir em comparação com as atividades contrárias, como os jogos (ESTADO DE MINAS, 2023). Tal atividade, traz ao cotidiano das crianças uma gama de estímulos, pondo-as num comodismo, devido a recorrentes atividades superficiais e pouco desafiadoras, levando a um baixo rendimento e interesse nas aulas (G1, 2024).

Considerando os fatos, como uma plataforma de jogos-sérios pode amenizar o baixo rendimento escolar?

2.2 Tecnologias utilizadas

Para a confecção do Bloomy, foram usadas tecnologias pensadas para a solução do problema proposto. A seguir, teremos o referencial teórico sobre essas ferramentas que serão utilizadas.

2.2.1 Design Thinking

De acordo com Talita Pagani, o design tem a função de desenhar um caminho para resolução de um problema — de maneira humana e com atenção individual acima das ferramentas de resolução (PAGANI, 2018).

Como visto na Revista eletrônica de biblioteconomia e ciência da informação, Design Thinking é um método de inovação com a função de entender e solucionar problemas de forma criativa em diversas condições. (FEDERAL DE SANTA *et al.*, 2022).

É apontado no livro *Design Thinking: Inovação em Negócios*:

Ao desafiar os padrões de pensamento, comportamento e de sentimento 'Design Thinkers' produzem soluções que geram novos significados e que estimulam os diversos aspectos (cognitivo, emocional e sensorial) envolvidos na experiência humana. (VIANNA *et al.*, 2012, p. 14).

Em serviços que lidam com pessoas, o modelo tradicional de resolução de problemas, sistemático e objetivo, não é adequado, é necessário se moldar às necessidades, emoções e desejos de cada indivíduo para entrega de requisitos (PAGANI, 2018).

2.2.2 Unified Modeling Language (UML)

É uma ferramenta de modelagem, que visa analisar e projetar sistemas orientados a objeto, limitando e direcionando os objetivos do projeto, buscando atender o melhor resultado em todas as áreas possíveis (BOOCH; RUMBAUGH; JACOBSON, 2006).

Segundo Guedes (2009), a UML possui o objetivo de auxiliar os engenheiros de software para a construção, caracterização e detalhamento da aplicação, não sendo exclusiva à apenas eles.

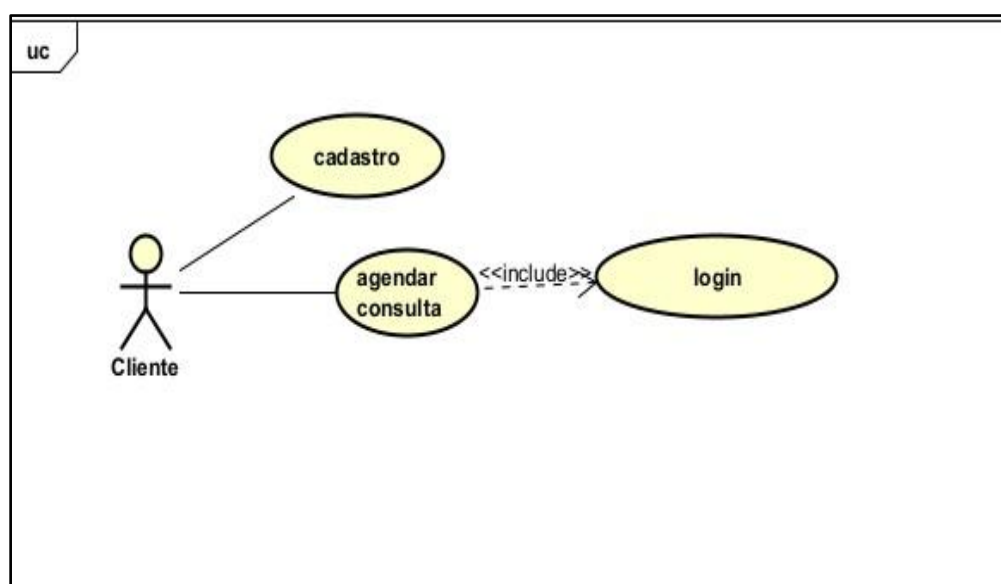
Conforme descrito por Larman (2000), essa ferramenta oferece padrões, que torna possível a visualização e disposição de soluções mais fácil — não prescreve passo a passo, mas oferece estrutura e seriedade ao seu projeto, tornando-o manutenível. Dentro desse modelo, temos padrões normativos para cada especificação do projeto — não necessariamente equivalentes ao código, mas aproximados — não sendo um modelo rígido, apenas um norte do que deve ser feito (FOWLER, 2007).

2.2.2.1 Diagrama de Casos de Uso

Assim como outros diagramas dinâmistas informais, o de Casos de Uso modela a atitude do sistema — como ele deve agir, que comportamento ele precisa ter — envolvendo contexto, e requisitos da aplicação (BOOCH; RUMBAUGH; JACOBSON, 2006).

Como apontado por Guedes (2009), objetiva exemplificar as funcionalidades do sistema, moldando de forma geral, o que ocorrerá — a cadeia de eventos vinculada a cada usuário e função — assim baseando outros diagramas. Ou seja, converge um ator e os processos que ele desencadeia na aplicação, baseando a construção do software — sendo atores os usuários e os casos de uso às funções — de fácil entendimento prévio da narração dos requisitos (LARMAN, 2000).

Figura 01 — Exemplo Diagrama de Caso de Uso



Fonte: Autoria Própria (2025).

O exemplo acima, demonstra o modelo simples de casos de uso de uma clínica, tendo um ator chamado “Cliente”, e os casos de uso “Cadastro”, “Agendar Consulta” e “Login”. O Cliente pode interagir com todos os processos, ou seja, ele pode se cadastrar, e em seguida gerenciar seu perfil; é possível que o ator “Cliente” agende uma consulta, mas apenas após efetuar o login, como mostra pela seta tracejada “<<include>>” — elemento condiciona uma ação antes que o caso de uso principal seja realizado.

2.2.2.2 Documentação dos Casos de Uso

Para Booch, Rumbaugh e Jacobson (2005), os comportamentos dos Casos de Uso é o conjunto de ações que são utilizadas para visualizar e documentar o sistema e em função do que foi estimado, convertendo tais requisitos em detalhes fixos.

Contudo, Guedes (2009) afirma que a documentação do caso de uso, é a maneira de simplificar as funções gerais expostas no diagrama — tendo o contexto dos processos, os atores que interagem, e as funções que irão ocorrer.

Afirma Larman (2000), que é um documento narrativo formal e sucinto — tem as mesmas funções que o diagrama derivado, só que de maneira detalhista, onde aponta as finalidades, visão geral e tipo do caso de uso utilizado.

Tabela 1 — Exemplo da Documentação do Caso de Uso

Nome do caso de uso	Marcar consulta.
Ator principal	Cliente.
Atores secundários	Nenhum.
Resumo	Este caso de uso expressa o fluxo de um cliente marcando uma consulta.
Pré-condições	É preciso efetuar o log-in.
Pós-condições	Uma consulta é cadastrada vinculada ao cliente,
Fluxo principal	
Ações do ator	Ações do Sistema
1. Solicitar log-in.	
	2. Inserir informações do cliente.
3. Realizar verificação de login.	4. Consultar cliente com base na senha fornecida e CPF.

5. Solicitar página de marcar consulta.	
6. Passa parâmetros para efetuar consulta.	
	7. Inserir consulta.
Restrições/ validações	<ol style="list-style-type: none"> 1. Para se cadastrar é preciso ser maior de idade. 2. Para marcar uma consulta, é preciso efetuar login.

Fonte: Autoria Própria (2025).

No exemplo, define-se o nome do caso de uso, chamado “Marcar Consulta”.

Então é informado o caso de uso geral, no entanto não há nenhum, então não fora especificado.

Logo, frisado o ator principal e o secundário, respectivamente, é aquele que mais interage com o caso de uso documentado, sendo ele o “Cliente”; não possui, logo não foi especificado. E também um resumo sobre a função desse caso de uso em específico.

Em seguida é definido as pré-condições e pós-condições, as etapas que ocorrem anterior e seguinte ao caso de uso, sendo elas “Efetuar login” e “Cadastrar Consulta” subsequentemente.

Após isso, temos o fluxo principal — onde mostra a cadeia de eventos principais que regem o caso de uso, separado por etapas dos atores e do sistema — como no exemplo, o login é solicitado pelo “Cliente”, que envia suas informações para o sistema inserir e fazer a verificação; com isso é possível fazer o processo de

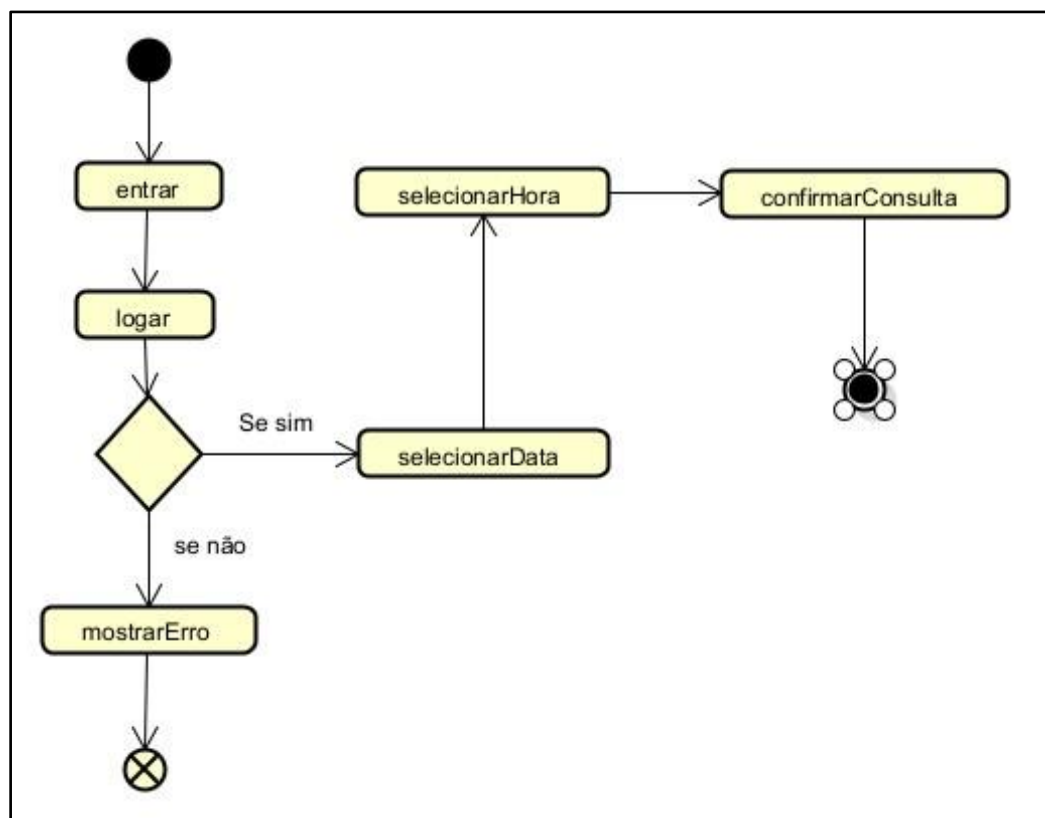
cadastrar consulta, onde o “Cliente” novamente informa seus dados para cadastrar e o sistema além de inserir, retorna à consulta finalizada.

2.2.2.5 Diagrama de Atividade

Fowler (2007) expõe, é a ferramenta que indica o fluxo — espécie de fluxograma, onde registra a cadeia de atividades lógicas — simultâneas ou intercaladas, que ocorrerão durante o uso do sistema.

Elucide Guedes (2009), que este modelo enfatiza o curso das operações que serão realizadas pelos métodos — ou seja, um conjunto de ações que expressam as funcionalidades, podendo ou não finalizar a atividade.

Figura 02 — Exemplo Diagrama de Atividade



Fonte: Autoria Própria (2025).

No exemplo acima, o diagrama se inicia com o círculo preto preenchido. segue para as atividades “entrar” e “login”, onde ocorre uma validação, se “login” retorna falso, o sistema mostrar invalidez e fecha atividade; caso retorne verdadeiro, o fluxo continua possibilitando marcar consulta, pelo caminho “selecionarData” e

“selecionarHorario”, podendo assim “confirmarConsulta” e encerrar as atividades como um todo.

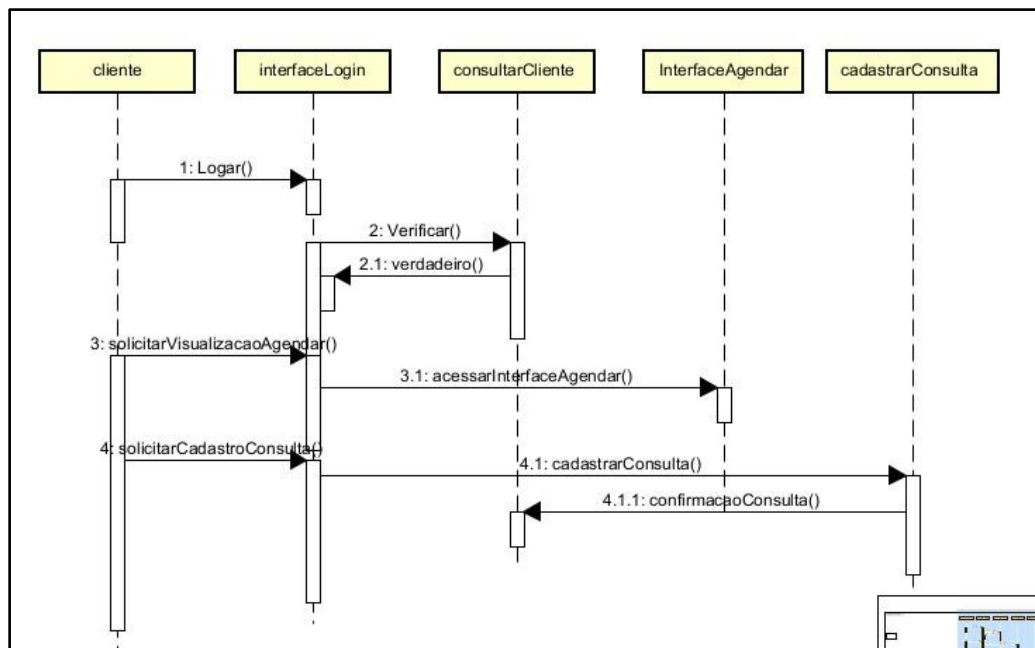
2.2.2.3 Diagrama de Sequência

Segundo Fowler (2007), é a fragmentação de um caso de uso baseada em cenários — o que acontecerá para o usuário em determinada situação — mostrando possíveis resultados para cada situação.

Como explicado por Guedes (2009), é apoiado nos processos desencadeados pelo usuário, determinando o desenrolar do cenário — validando e complementando o diagrama de classes, possibilitando uma aplicação mais sólida e precisa.

Com uma ordenação de mensagens e eventos, o diagrama de sequência dispõe os objetos que reagem e afetam outros, prevendo a duração da interação de usuário com objetos, chamadas de funções e valores de retorno (BOOCH; RUMBAUGH; JACOBSON, 2006).

Figura 03 — Exemplo Diagrama de Sequência



Fonte: Autoria Própria (2025).

Podemos ver neste exemplo, os retângulos de ativação, atores ou funcionalidades ativas no diagrama, sendo eles “Cliente”, “InterfaceLogin”, “consultarCliente”, “interfaceAgendar” e “cadastrarConsulta”. O “Cliente”, na sua linha de vida – linha tracejada abaixo dos retângulos — pode efetuar o login, para ser

encaminhado para “interfaceLogin”. Após essa solicitação, os dados fornecidos são levados para “consultarCliente”, onde as informações são validadas, para que caso seja verdadeiro, ou seja, esse usuário está cadastrado no sistema, é possível ver outras funcionalidades.

O “Cliente” pode navegar pela “interfaceAgendar”, onde é cabível o cadastro de consulta, que retornará o resultado “confirmacaoConsulta”.

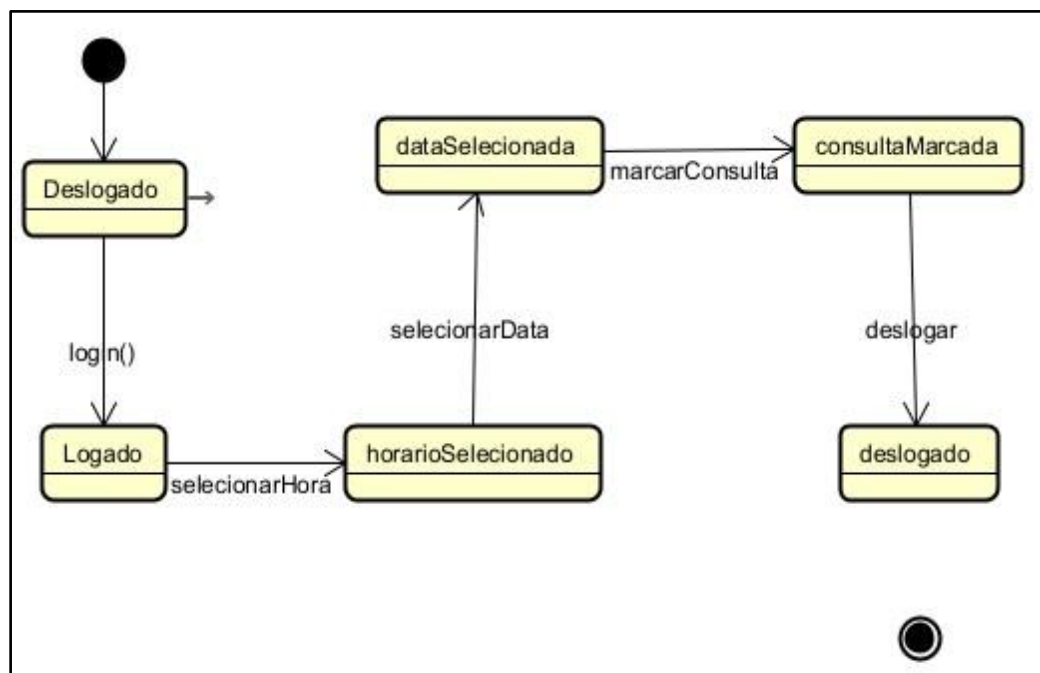
2.2.2.4 Diagrama de Máquina de Estados

De acordo com Fowler (2007), é um diagrama que explana as mudanças de estados em uma máquina — seja devido a imposições ou condições — também definindo uma base de dependência do seu objeto sobre o estado atual.

Usado para modelar as atividades dinâmicas de cada aspecto individual da vida de um sistema, como uma cadeia de eventos determinados pelo estado do objeto — que causará uma ou mais reações (BOOCH; RUMBAUGH; JACOBSON, 2006).

Larman Craig (2000), explica que o diagrama de estados modela apenas o que é necessário, indo de detalhista à simplista de maneira arbitrária — de acordo com os requisitos do sistema.

Figura 04 — Exemplo Diagrama de Máquina de Estados



Fonte: Autoria Própria (2025).

É exemplificado o início do diagrama, marcado pelo círculo preto. Como explicado anteriormente, esse diagrama demonstra estados — retângulos amarelos — que representam a reação após ações das classes e objetos, sendo eles: “Deslogado”, “Logado”, “horarioSelecioneado”, “dataSelecioneada”, “consultaMarcada” e novamente “Deslogado”. As setas indicam eventos, como métodos são ações que afetam os estados, sendo elas “login”, “slecionarHora”, “selecionarData”, “marcarConsulta” e “deslogar”. O diagrama é finalizado pelo círculo preto dentro de outro.

2.2.3 Wireframes

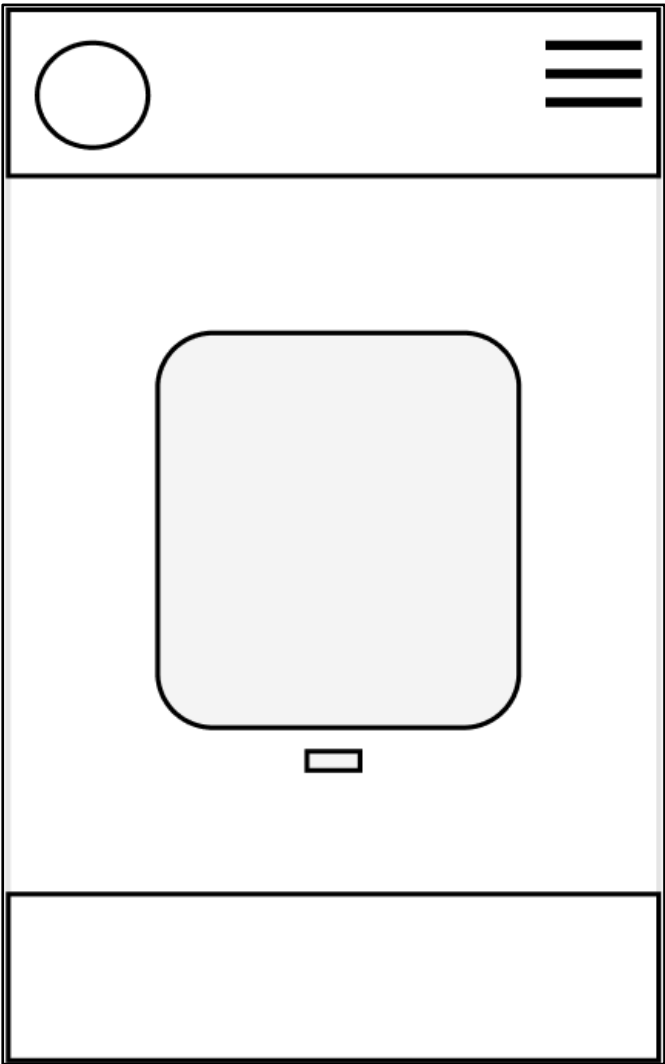
Exemplifica Caio Oliveira, (2021) que *wireframes* são como as interfaces irão ser montadas — representam a aparência, navegação e funcionalidades básicas do site — tornando mais concretos os eventos feitos em diagramas.

Como desenhos, afirma Fabrício (2014), os *wireframes* podem ser mais ou menos elaborados, depende de quão perto do design final o produto se encontra, dito isso – *wireframes* de baixa fidelidade são como rascunhos, passíveis de mudanças.

De acordo com Rogério (2018), os *wireframes* de alta fidelidade, são os mais próximos de um resultado — quanto mais específico for, menos chances de explorar criativamente se tem, abaixando as possibilidades de uma identidade visual própria.

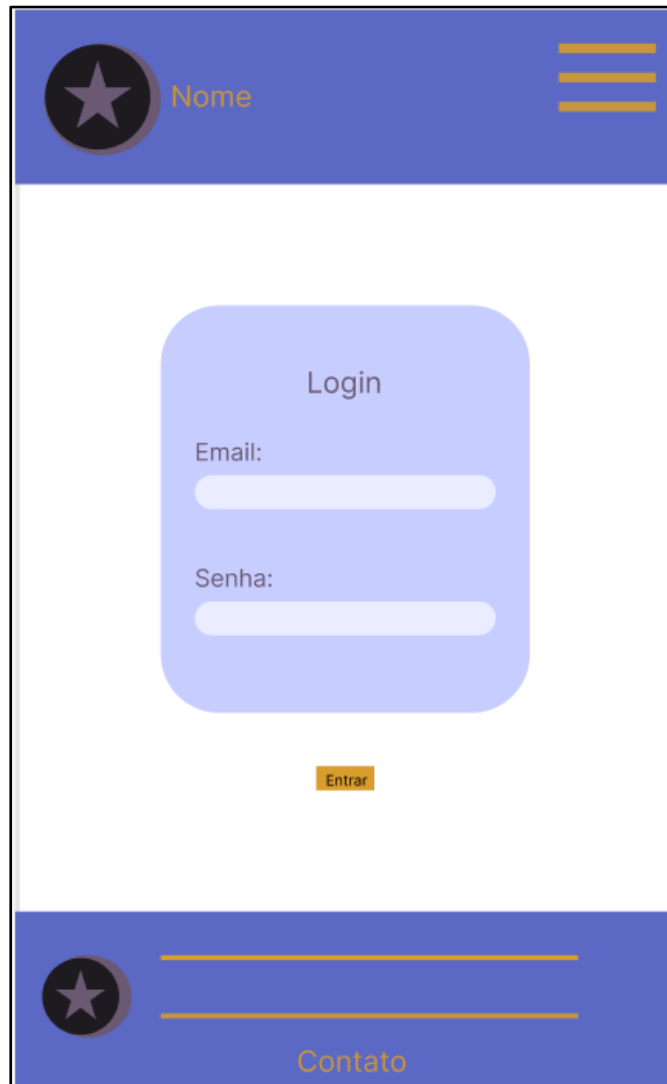
Para modelar interfaces, pode ser utilizado o Figma, um *software* que auxilia no processo de desenvolvimento em *wireframes*, na aparência e na navegabilidade (ACETI *et. al*, 2023).

Figura 05 — Exemplo Wireframe de Baixa Fidelidade



Fonte: Autoria Própria (2025).

Figura 06 — Exemplo Wireframe de Alta Fidelidade



Fonte: Autoria Própria (2025).

2.2.4 HTML

De acordo com Fábio Flatschart (2011), o HTML, ou Marcação de Linguagem de Hipertexto (tradução de *HyperText Markup Language*), é a principal linguagem para construção de páginas na internet.

Segundo Maurício Samy (2008), Hipertexto é todo conteúdo que pode ser inserido em páginas da internet e que possuam a possibilidade de se interligarem com outras aplicações da *web*, utilizando de *links* para a sua construção.

Na linguagem HTML, utilizamos de *tags* para a escritura da linguagem, que usam das sintaxes “< >” ao início e “</ >” ao fim de cada linha. Na construção de

aplicações web, as *tags* servem para estruturar como os elementos são inseridos na página (FLATSCHART, 2011).

As principais *tags* utilizadas no HTML para a construção de aplicações são:

- “<!DOCTYPE html>” informa a versão utilizada na marcação ao navegador, permitindo que o mesmo processe a página da melhor forma na internet;
- “<html> & </html>” criado logo após a tag *DOCTYPE*, esta tag detém todo o código da página, indicando o início e o fim do mesmo;
- “<head> & </head>” está localizado dentro da *tag* <html>, serve como cabeçalho da aplicação, além de deter as configurações da página;
- “<body> & </body>” está localizado dentro da *tag* <html>, todo o código dos elementos visíveis na página é escrito dentro desta *tag*.

A seguir demonstraremos um código em HTML, sendo ele uma interface simples, que exemplifica a construção de um formulário:

Figura 07 — Exemplo Código em HTML

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5    <title>Exemplo</title>
6  </head>
7
8  <body>
9    <div>
10      <h1>Formulário</h1>
11      <hr/>
12      <form>
13        <label>Nome:</label>
14        <input/>
15
16        <br/>
17
18        <label>Email:</label>
19        <input/>
20
21        <br/>
22
23        <label>Senha:</label>
24        <input/>
25
26        <br/>
27
28        <button>Enviar</button>
29      </form>
30    </div>
31  </body>
32
33 </html>
```

Fonte: Autoria Própria (2025).

Linha 5: É demonstrado a tag “<title>”, que define o título da página.

Linha 8: Cria uma divisória entre as outras *tags* por meio da <div>.

Linha 10: Define o tamanho “<h1>” ao texto “Formulário”.

Linha 11: Demarca uma diferença entre a sessão pelo <hr>.

Linha 12: Expressa o início de um formulário.

Linha 13: Define uma legenda para algum campo pela *tag* “<label>”, como “Nome:”, “Email:” e “Senha:”.

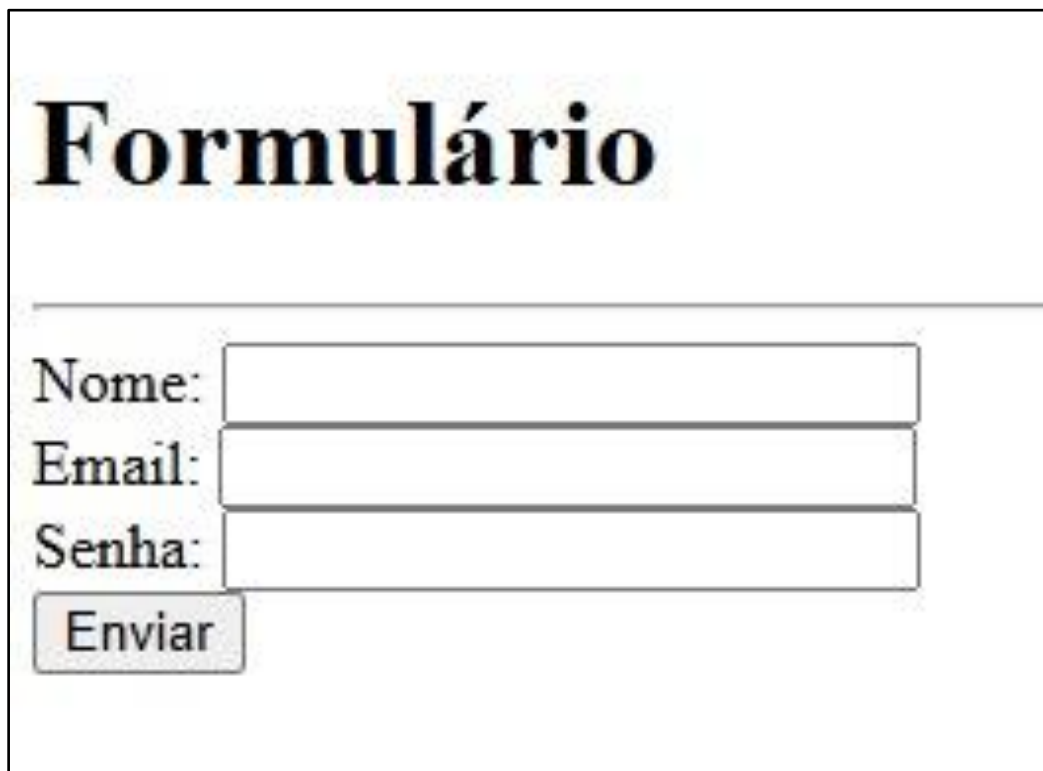
Linha 14: A tag “<input>” expressa um elemento de campo para inserir informações

Linha 16: Dá um espaçamento entre os elementos pela *tag* “
”.

Linha 28: Inicia um elemento botão chamado “Enviar” pela *tag* “button”.

A seguir o resultado do exemplo, executado no navegador:

Figura 08 — Exemplo Formulário em HTML



Formulário

Nome:

Email:

Senha:

Fonte: Autoria Própria (2025).

2.2.5 CSS

Folhas de Estilo em Cascata, é a linguagem de estilização utilizada para a formatação do *layout* de páginas. Ele facilita a estilização, generalizando o código em um arquivo único — economizando tempo de maneira organizada. (JOBSTRAIBIZER, 2009).

Segundo Evandro Miletto e Silvia Bertagnolli (2014), o CSS possibilita a criação de personalizada para títulos, imagens, listas e outros; além da criação de paleta de cores, fontes para texto, alinhamento dos objetos, e outras características relacionada à aparência da aplicação.

Afirma Felipe Scheidt (2015) que o termo “Folha de Estilo” significa um conjunto de regras, que definem a aparência de uma aplicação *web* — onde as características da página como cores e posições dos objetos são definidas por regras.

A seguir, temos um exemplo de como estilizamos o formulário em HTML que fizemos anteriormente:

Figura 9 — Exemplo Código em CSS

```
1  body {  
2      display: flex;  
3      justify-content: center;  
4      font-family: sans-serif;  
5  }  
6  
7  .caixaFormulario {  
8      width: 30%;  
9      padding: 10px;  
10     border-radius: 10px;  
11     background-color: lightblue;  
12     text-align: center;  
13 }  
14  
15 .entrada {  
16     border: none;  
17     margin-bottom: 10px;  
18     border-radius: 5px;  
19     width: 100%;  
20     height: 20px;  
21 }  
22  
23 .botao {  
24     border: none;  
25     padding: 5px;  
26     border-radius: 5px;  
27 }  
28
```

Fonte: Autoria Própria (2025).

Linha 1: A tag “body” é aberta, onde definimos a estilização geral dos elementos da página.

Linha 2: Definimos o alinhamento dos objetos da página como flexíveis.

Linha 3: Centraliza os elementos dentro do corpo da página.

Linha 4: É definida qual a fonte padrão da página.

Linha 7: Aqui iniciamos a definição de estilização de elemento com a classe “caixaFormulario”. O mesmo acontece na linha 15 e 23, apenas com classes diferentes.

Linha 8: É definida em 30% a largura dos elementos com a classe.

Linha 9: Adicionamos um espaçamento interno de 10 pixels ao redor do conteúdo do elemento.

Linha 10: Arredonda as extremidades do elemento em 10 pixels.

Linha 11: Definimos uma cor de fundo para os elementos da classe, sendo a cor escolhida o azul claro.

Linha 12: Centraliza o texto do elemento.

Linha 16: É removido qualquer borda padrão de elementos da classe “entrada”.

Linha 17: Adicionamos uma margem inferior de 10 pixels no elemento, criando um espaçamento abaixo do mesmo.

Linha 18: Como na linha 10, definimos o arredondamento das bordas do elemento.

Linha 19 e 20: É definida, respectivamente, a largura e altura do elemento.

Figura 10 — Exemplo Código em HTML e CSS

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5    <title>Exemplo</title>
6    <link rel="stylesheet" href="exemplo.css">
7  </head>
8
9  <body>
10   <div class="caixaFormulario">
11     <h1>Formulário</h1>
12     <hr/>
13     <form>
14       <label>Nome:</label>
15       <input class="entrada"/>
16
17       <br/>
18
19       <label>Email:</label>
20       <input class="entrada"/>
21
22       <br/>
23
24       <label>Senha:</label>
25       <input class="entrada"/>
26
27       <br/>
28
29       <button class="botao">Enviar</button>
30     </form>
31   </div>
32 </body>
33
34 </html>
```

Fonte: Autoria Própria (2025).

No código HTML, conectamos o arquivo CSS através da tag “<link>”, permitindo com que a estilização que definimos seja aplicada aos elementos construídos na tela.

Figura 11 — Exemplo Formulário em CSS

A imagem mostra um formulário web estilizado com CSS. O formulário tem um fundo azul claro e uma borda arredondada. No topo, o título "Formulário" está em uma fonte grande e preta. Abaixo do título, há três campos de entrada de texto, cada um precedido por um rótulo: "Nome:", "Email:" e "Senha:". Os campos de entrada são retângulos brancos com bordas arredondadas. No final do formulário, há um botão "Enviar" com um fundo cinza claro e uma borda arredondada.

Fonte: Autorial Própria (2025).

2.2.6 JavaScript

Baseado em David Flanagan (2013), o *JavaScript* (JS) é utilizado com frequência em sites e navegadores modernos — sendo uma das linguagens que o desenvolvedor web deve conhecer, já que especifica o comportamento das páginas na internet, dando movimento.

Na programação, é uma linguagem dinâmica, possibilitando manipular funções de forma personalizada, criando e automatizando no sistema, o que torna o JS uma linguagem para profissionais e leigos no assunto (SILVA, 2010).

Para a construção de um site, o *JavaScript* é representado pela tag “<script>”, geralmente localizada dentro da tag “<head>”, permitindo a associação de arquivos externos ao HTML, mas sempre apontam para arquivos em JS (PUREWAL, 2014).

Agora, adicionaremos esses conceitos no formulário anterior de HTML e CSS, para adicionar funcionalidades:

Figura 12 — Exemplo Código em JS

```
<script>
  function mensagemDeEnvio() {
    var nome = document.getElementById("entradaNome");
    var email = document.getElementById("entradaEmail");
    var senha = document.getElementById("entradaSenha");

    alert(
      "Nome do usuário=" + nome.value +
      "\nEmail do usuário=" + email.value +
      "\nSenha do usuário=" + senha.value
    );
  }
</script>
```

Fonte: Autoria Própria (2025).

Linha 1: Inicia a *tag* “<script>”, onde fica o conteúdo do código de *JavaScript*.

Linha 2: É definida uma função chamada “mensagemDeEnvio”, que será executada ao clicar no botão.

Linha 3 a 5: São criadas variáveis que armazenam o conteúdo dos elementos com os Ids especificados em cada linha, como o “entradaNome”.

Linha 7: Quando o código for executado, a tag “alert(” abrirá uma caixa de alerta no seu navegador e mostrará como uma mensagem o conteúdo dentro da tag.

Linha 8 a 10: Mostra o texto escrito concatenado ao valor armazenado da variável, o qual é obtido através do “.value”. O “\n” serve para quebrar a linha antes do texto aparecer, impedindo que tudo fique na mesma linha.

Figura 13 — Exemplo Código em JS e HTML

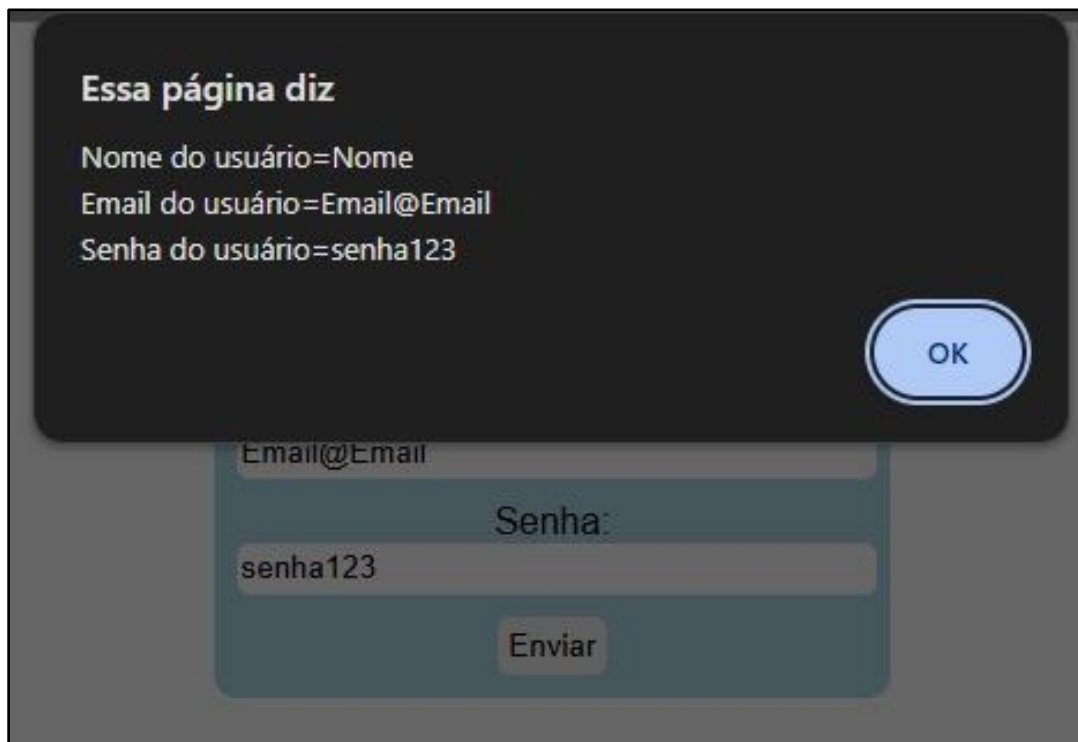
```
<button type="button" onclick="mensagemDeEnvio()" class="botao">Enviar</button>
```

Fonte: Autoria Própria (2025).

Para que o botão realmente execute a função, colocamos no “onclick=” a nossa função, nesse caso a “mensagemDeEnvio()”, conectando o botão com a mesma e tornando ela executável.

A seguir, veja a caixa de alerta que a função abre quando apertamos o botão:

Figura 14 — Exemplo Formulário em JS



Fonte: Autoria Própria (2025).

2.2.7 React

React é uma biblioteca de *JavaScript* fundamentado pela Meta, sendo extremamente popular, de código aberto e frequentemente atualizada, seu principal foco é construir interfaces de usuário com atualizações rápidas e reativas (NEVES, 2023).

A construção de telas pelo *React* é feita com componentes e elementos, tendo uma segmentação maior do código. Suas interações são baseadas em estados e eventos, assim gerando códigos reativos e dinâmicos (AZZOLINI, 2021).

React é especificamente utilizado para criar interfaces de usuário com interações reativas e de atualização imediata, seu uso se tornou imensamente popular no desenvolvimento *web*, e, hoje, é um dos frameworks mais utilizados (SILVA; OLIVEIRA, 2023).

Abaixo temos a construção de um formulário em *React* para exemplificar os conceitos:

Figura 15 — Exemplo código em React

```
1  import './App.css'
2
3  function App() {
4
5    return (
6      <>
7
8        <div className='form'>
9          <h1>Formulário</h1>
10         <hr />
11         <form>
12           <label>Nome:</label>
13           <input />
14
15           <br />
16
17           <label>Email:</label>
18           <input />
19
20           <br />
21
22           <label>Email:</label>
23           <input />
24
25           <br />
26
27           <button>Enviar</button>
28         </form>
29       </div>
30     </>
31   )
32 }
33
34
35 export default App
```

Fonte: Autoria Própria (2025).

O código acima é muito parecido com o código do formulário simples feito em HTML. Além de parte da estrutura ser diferente, como a ausência das tags “<head>” e “<body>”, o CSS é conectado através da tag “import” e a construção da página é feita dentro de uma função, neste caso a nomeada de “App()”.

Figura 16 — Exemplo Formulário em React



Formulário

Nome:

Email:

Email:

Enviar

Fonte: Aatoria Própria (2025).

2.2.8 React Native

Embora *React Native* tenha suas semelhanças com *React*, isso não descarta suas diferenças, a linguagem é executada com o *JavaScript* Core com uma ponte para tradução do código para renderização em qualquer sistema operacional móvel (SILVA; DE SOUZA, 2019).

Como apontado por Bruna Escudelar e Diego Pinho, *React Native* é um *framework* que permite aplicativos híbridos mobile sejam desenvolvidos com as mais sofisticadas ferramentas *front-end*, com uma compatibilidade com iOS e *Android* (ESCUDELARIO; PINHO, 2021).

Segundo Franklyn Seabra e Rogério Bezerra, *React Native* não funciona como um sistema *web* em *WebView*, e sim como um elemento nativo do sistema do dispositivo, assim se desempenhando como um aplicativo na camada nativa (BEZERRA, 2021).

A diante, adicionaremos o *framework React Native* no formulário anterior, veja abaixo:

Figura 17 — Exemplo de Código em React Native

```
1  import { FormBasicExample } from '@components/FormBasicExample';
2
3  export default function FormScreen() {
4    |   return <FormBasicExample />;
5  }
```

Fonte: Autoria Própria (2025).

Esse trecho existe para carregar e mostrar o componente “FormBasicExample”, onde está localizado o formulário básico que será exibido. Esse código pode ser bem útil para a organização do projeto, principalmente quando estamos utilizando rotas.

Figura 18 — Exemplo de Código em React Native

```
1  import { useState } from 'react';
2  import { Alert, StyleSheet, TextInput, View } from 'react-native';
3
4  import { ThemedText } from '@components/ThemedText';
5  import { ThemedView } from '@components/ThemedView';
6
```

Fonte: Autoria Própria (2025).

Na imagem acima, estão sendo feitas importações para o componente de *React Native*, o preparando para a utilização de estados, criação de alertas, campos de texto e *layouts*, aplicação de estilos e utilização de componentes personalizados.

Figura 19 — Exemplo de Código em React Native

```
16   return (  
17     <ThemedView style={styles.caixaFormulario}>  
18       <ThemedText type="title">Formulário</ThemedText>  
19       <View style={styles.hr} />  
20       <View>  
21         <ThemedText>Nome:</ThemedText>  
22         <TextInput  
23           style={styles.entrada}  
24           value={nome}  
25           onChangeText={setNome}  
26           placeholder="Digite seu nome"  
27         />  
28  
29         <ThemedText>Email:</ThemedText>  
30         <TextInput  
31           style={styles.entrada}  
32           value={email}  
33           onChangeText={setEmail}  
34           placeholder="Digite seu email"  
35           keyboardType="email-address"  
36           autoCapitalize="none"  
37         />  
38       </View>  
39     </ThemedView>  
40   );  
41 }
```

Fonte: Autoria Própria (2025).

O código acima, define os elementos da interface do formulário, montando os campos “nome” e “email”, que são estilizados com tema, ligados a estados internos que são atualizados em tempo real conforme o usuário digita.

Figura 20 — Exemplo de Código em React Native

```
export function FormBasicExample() {  
  const [nome, setNome] = useState('');  
  const [email, setEmail] = useState('');  
  const [senha, setSenha] = useState('');  
  
  function mensagemDeEnvio() {  
    Alert.alert('Formulário enviado', `Nome: ${nome}\nEmail: ${email}\nSenha: ${senha}`);  
  }  
}
```

Fonte: Autoria Própria (2025).

Esse código cria três estados para capturar o que o usuário digita no formulário e uma função que exibe esses dados em um alerta quando o formulário é enviado.

Figura 21 — Exemplo de Código em React Native

```
38
39     <ThemedText>Senha:</ThemedText>
40     <TextInput
41       style={styles.entrada}
42       value={senha}
43       onChangeText={setSenha}
44       placeholder="Digite sua senha"
45       secureTextEntry
46     />
47
48     <View style={{ marginTop: 10 }}>
49       <ThemedText
50         style={styles.botao}
51         onPress={mensagemDeEnvio}
52         selectable={false}
53         accessibilityRole="button"
54       >
55         Enviar
56       </ThemedText>
57     </View>
58   </View>
59 </ThemedView>
60 );
61 }
```

Fonte: Autoria Própria (2025).

O trecho acima, adiciona o campo de senha ao formulário e cria um botão estilizado que, ao ser pressionado, chama a função “mensagemDeEnvio” para mostrar os dados digitados em um alerta.

Figura 22 — Exemplo de Código em React Native

```
62
63 const styles = StyleSheet.create({
64   caixaFormulario: {
65     width: '90%',
66     maxWidth: 400,
67     padding: 16,
68     borderRadius: 10,
69     backgroundColor: 'lightblue',
70     alignSelf: 'center',
71     marginTop: 32,
72     textAlign: 'center',
73   },
74   hr: {
75     borderBottomWidth: StyleSheet.hairlineWidth,
76     borderBottomColor: '#888',
77     marginVertical: 12,
78   },
79   entrada: {
80     borderWidth: 0,
81     marginBottom: 10,
82     borderRadius: 5,
83     width: '100%',
84     height: 40,
85     backgroundColor: '#fff',
86     paddingHorizontal: 8,
87   },
88   botao: {
89     backgroundColor: '■ #87ceeb',
90     borderRadius: 5,
91     padding: 10,
92     textAlign: 'center',
93     color: '#222',
94     fontWeight: 'bold',
95     overflow: 'hidden',
96   },
97 });
```

Fonte: Autoria Própria (2025).

Este código é responsável pela estilização dos componentes da página, criando estilos para o container principal, os campos de texto e o botão.

Figura 23 — Exemplo Formulário em React Native

The image shows a mobile application interface with a light blue header bar containing the title "Formulário". Below the header, there are three input fields labeled "Nome:", "Email:", and "Senha:". Each field is a white rectangle with rounded corners. Below the "Senha:" field is a blue button with the text "Enviar" in white. The background of the app is a light gray. At the bottom, there is a white navigation bar with three icons: a house icon labeled "Ho...", a play button icon labeled "Explore", and a right-pointing arrow icon labeled "Formulá...". The status bar at the top shows the time "22:45" and various system icons including Bluetooth, signal strength, and battery level.

Fonte: Autoria Própria (2025).

2.2.9 TypeScript

TypeScript é uma linguagem de programação desenvolvida com base no *JavaScript* padrão, adicionando recursos avançados para a linguagem. Foi projetada

com o principal intuito de solucionar limitações que o *JavaScript* possui (MORÓRÓ, 2024).

De acordo com Francisco Souza, Edilson Lima e Elda Caridade (2022), essa linguagem é fortemente tipada e mais escalável que o JS — sendo muito utilizada por *frameworks* de *FrontEnd* — algo benéfico para desenvolvedores *FullStack*.

Assim, demonstraremos a seguir a implementação do *TypeScript* no formulário de exemplo:

Figura 24 — Exemplo Código em TS

```
1  function mensagemDeEnvio(): void {
2      const nome = document.getElementById("entradaNome") as HTMLInputElement;
3      const email = document.getElementById("entradaEmail") as HTMLInputElement;
4      const senha = document.getElementById("entradaSenha") as HTMLInputElement;
5
6      alert(
7          "Nome do usuário=" + nome.value +
8          "\nEmail do usuário=" + email.value +
9          "\nSenha do usuário=" + senha.value
10     );
11 }
12
```

Fonte: Autoria Própria (2025).

Linha 1: Declaramos uma função chamada “mensagemDeEnvio”. A tag “void” significa que esta função não retorna nenhum valor.

Linha 2 a 4: São declaradas as constantes “nome”, “senha” e “email”, que parecido com o *JavaScript*, pegam o valor dos elementos do HTML através dos seus Ids. A tag “HTMLInputElement” presente nas linhas é uma asserção de tipo no *TypeScript*, onde diz que aquele elemento é um campo de entrada de texto.

Linha 6: A tag “alert(” funciona em *TypeScript* do mesmo jeito que no *JavaScript*, abrindo uma caixa de alerta quando executada.

Linha 7 a 9: Assim como no *JavaScript*, apresenta o texto armazenado na constante.

2.2.10 Banco de dados

De acordo com C.J. Date (2004), um sistema de Banco de Dados é apenas uma coleção computadorizada de gerenciamento de registros, servindo como um repositório para arquivos de dados digitais.

Os bancos de dados possuem um grande impacto em áreas que utilizam de computadores, pois sua capacidade de abrigar dados de qualquer tamanho e complexidade torna os bancos benéficos ao trabalho (ELMASRI; NAVATHE, 2011).

O banco de dados é quem molda a interface, afirma Lev Manovich (2015); a estrutura de armazenamento de dados é quem dita como deverá ser o funcionamento de uma página, definindo os elementos que precisaram ser criados para aquela tela.

2.2.11 Firebase

De acordo com Machado (2021), é uma plataforma que possibilita consulta e manipulação de dados em nuvem, com o adicional de escalabilidade, segurança ao banco, bibliotecas de autenticação e até gerencia de dados em escala global.

O uso dessa ferramenta traz diversos benefícios — uma organização que permite fácil entendimento, escalabilidade para suportar grandes demandas, além de possuir formas de otimizar o desenvolvimento de *softwares* (ALURA, 2024)

Entre os serviços oferecidos pelo *Firebase*, estão incluídos um serviço de banco de dados chamado de *Cloud FireStore*, a autenticação de usuários, envio de notificações e armazenamento de arquivos, como dito por Amanda Anjolin Rodrigues (2021).

2.2.12 Construct

É uma plataforma intuitiva para o desenvolvimento de jogos 2D, utiliza de ferramentas como a Folhas de Eventos e *Layout*, como meio mais prático e dinâmico para projetos do ramo — ideal para leigos (PIMENTEL, 2025).

Como notado por Rodrigo Ribeiro Silva, Luis Rivero e Rodrigo dos Santos (2021), *Construct* é muito vantajoso para um desenvolvimento rápido dos jogos — pouco complexo, acessível, com uma comunidade de suporte e plataformas flexíveis para disponibilizar os jogos.

De muitas ferramentas de desenvolvimento de jogos, essa mostra uma simplicidade e didática na construção, facilitando assim o desenvolvimento e implementação de ideias na criação de videogames (ALVES; AZEVEDO; COSTA JUNIOR, 2019).

O *Construct* se mostra mais eficaz em prototipação e facilitando a elaboração em comparação a similares, tendo recursos mais adequados para uma equipe onde nem todos possuem o mesmo conhecimento técnico (CARVALHO; ARANHA; CARDIA NETO, 2022).

2.2.13 C++

A linguagem de programação C, a qual serviu de base para a construção da linguagem C++, foi inicialmente criada para conseguir ser interpretada por máquinas de forma eficiente e rápida (HORSTMANN, 2008).

Como descrito por Diego Rodrigues (2024), C++ surgiu como extensão da linguagem C, incluindo estruturas de programação orientada a objeto junto ao controle de baixo nível do C, formando assim uma linguagem muito versátil e poderosa.

Conforme Marques (2023), o C++ efetiva alta performance em diferentes sistemas operacionais, já que é compilada justamente na máquina, otimizando o código de maneira organizada e eficiente.

Abaixo você pode ver um código em C++, criado para ligar e desligar um LED ao apertar de um botão, a estrutura utilizada é específica para funcionar em microcontroladores.

Com isso em mente, construímos este exemplo abaixo, para possibilitar a visibilidade do código em C++:

Figura — Exemplo de código em C++

```
1  const int botao = 2; //Define a porta para o botão
2  const int led = 13; //Define a porta para o LED
3
4  int estadoDoBotao = 0;
5
6  void setup() {
7      pinMode(botao, INPUT);
8      pinMode(led, OUTPUT);
9  }
10
11 void loop() {
12     estadoDoBotao = digitalRead(botao);
13
14     if (estadoDoBotao == HIGH) {
15         digitalWrite(led, HIGH); //Liga o LED
16     }
17     else {
18         digitalWrite(led, LOW); //Desliga o LED
19     }
20 }
```

Fonte: Autoria Própria (2025)

Linha 1: É declarada uma constante de tipo inteiro armazenando o valor 2, de nome “botao”, esta linha se refere à porta do microcontrolador onde o botão está conectado.

Linha 2: É declarada uma constante de tipo inteiro armazenando o valor 13, de nome “led”, esta linha se refere à porta do microcontrolador onde o LED está conectado.

Linha 4: É declarada uma variável de tipo inteiro inicializada com o valor 0, de nome “estadoDoBotao”, esta linha receberá o estado atual do botão.

Linha 6: É criada uma função de nome “setup”, esta função é imediatamente executada ao código ser interpretado pela máquina.

Linha 7: A porta que o botão está conectado é definido como uma entrada.

Linha 8: A porta que o LED está conectado é definido como uma saída.

Linha 11: É definida uma função de nome “*loop*”, esta função será executada repetidamente e indeterminadamente.

Linha 12: A variável “estadoDoBotao” recebe o valor do botão.

Linha 14: É criado um operador lógico “*if*”, o código em seu interior será executado apenas se o botão estiver ligado.

Linha 15: Liga o LED.

Linha 17: É criado um operador lógico “*else*”, o código em seu interior será executado apenas se as condições para o último operador lógico “*if*” não forem atingidas.

Linha 18: Desliga o LED.

2.2.14 Internet of Things (IOT)

A internet das coisas — é um conceito, onde dispositivos físicos, por meio de sensores, se conectam à internet — sendo um facilitador no cotidiano atual, defende Eduardo Magrani (2018).

É apontada como uma evolução, onde dispositivos físicos podem ser manuseados para qualquer função, facilitando o cotidiano por meio de uma estrutura mais inteligente (SINCLAIR, 2018).

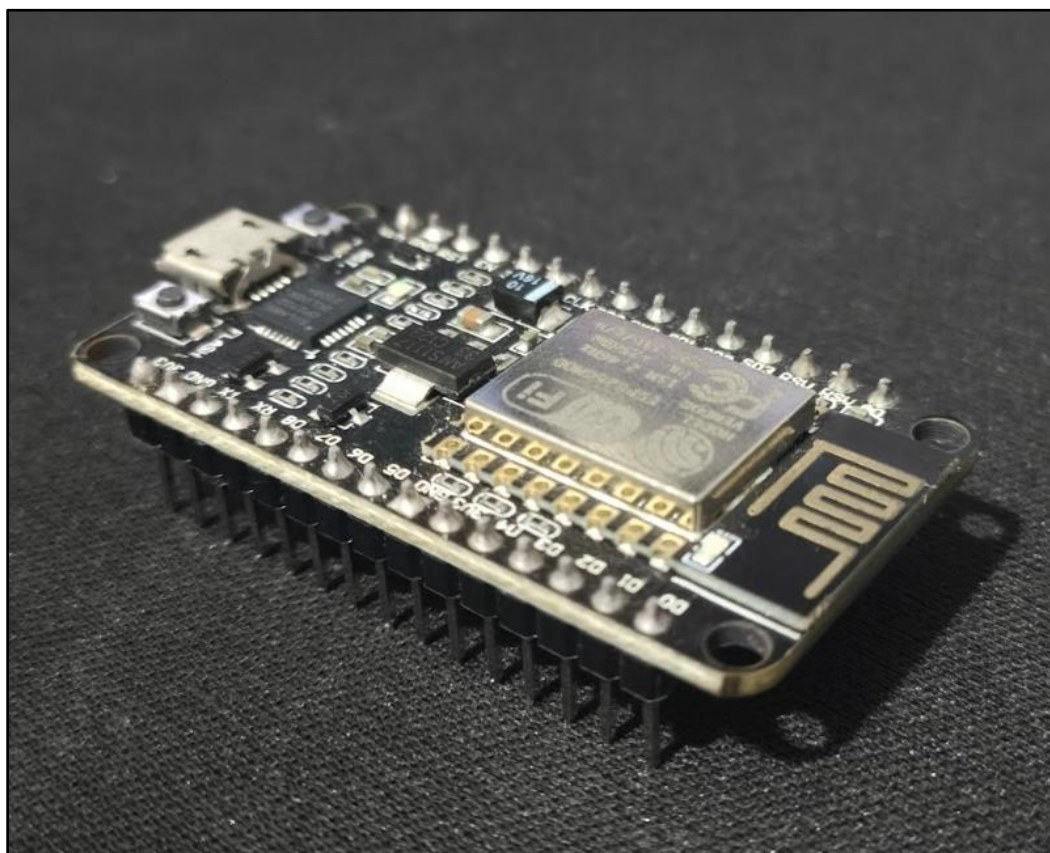
Segundo Corneto e Márcia (2024), a junção vinculada ao termo IoT, interage com o ambiente e com outros dispositivos, podendo coletar uma gama de dados, o que serve como base e meio para a criação de outros sistemas.

2.2.15 ESP32

De acordo com Pedro Bertoleti (2019), é aCPU (Unidade Central de Processamento) e a memória em uma pequena placa de circuito impresso que contém diversas capacidades de comunicação sem fio.

O ESP32 veio como uma versão mais poderosa do microcontrolador, já consolidado no mercado, ESP8266, tendo dois núcleos de processamento, mais periféricos e Bluetooth, continuando com um preço competitivo (MORAIS, 2023).

Figura 26 — Microcontrolador ESP8266



Fonte: Autoria Própria (2025).

De acordo com Allyson Nascimento *et. al* (2023), o microcontrolador ESP32, especialmente ESPWROOM-32, possui suporte para Wi-Fi, BLE (*Bluetooth Low Energy*) e *Ethernet*, além de possuir grande eficiência energética.

2.2.16 Modelagem 3D

A modelagem, como explicado por Priscila Argoud (2024), é uma técnica de recriar o mundo real de forma virtual, criando objetos tridimensionais (3D) nos meios digitais mimetizando a realidade ao aplicar conceitos físicos e geoespaciais.

Contudo, por meio da modelagem se busca o detalhamento durante a construção, possibilitando uma imersão ao usuário — trazendo o virtual para algo mais concreto e agradável (VILELA; LOPES; LIMA, 2015).

2.2.17 Impressão 3D

Como dito por Moisés Miranda Morandini e Gustavo Henrique Del Vecchio (2020), a impressão tridimensional — é a capacidade de modelar fisicamente, objetos

e seres inanimados, de diversos tipos — desde detalhados, simples, grandes ou pequenos.

É explicado por Eloar Froboese Silva *et. al* (2020), que o processo de impressão é feito em camadas — mais especificamente fatias baseadas no formato do modelo que está sendo impresso — que são adicionadas uma junta da outra, criando espessura e relevo.

Afirma Leonardo Gomez Castillo, que a impressão 3D, atrai alta atenção de quem a manuseia, apesar de complexas, ao transformar os modelos escaneados em atrações físicas, estimula bastante a criatividade e pensamento livre.

3 DESENVOLVIMENTO

Nesta seção do documento serão apresentadas etapas atingidas durante o planejamento, a documentação e resultados obtidos no desenvolvimento. Explanando os detalhes de todos os processos, criativos e técnicos, que consolidaram o sistema Bloomy.

3.1 Conceitos

Antes de iniciar a construção, foram planejados os conceitos fundamentais para confeccionar a solução. A Bloomy surgiu, a princípio como a plataforma que oferece atividades educacionais personalizáveis, utilizando do estímulo sensorial tátil e visual, para um público infantil específico — que adiante, identificou-se como alunos do Fundamental I. Contudo, uma abordagem menos específica, direcionou o foco do desenvolvimento, tornando o processo mais objetivo e conclusivo.

O conceito que mais se destaca, é o nome dado ao protótipo, vem da junção da palavra "*Bloom*", que significa florescer em inglês, com o pronome possessivo "*my*", empregando um parecer pessoal ao florescer — enfatizando o desenvolvimento individual que é objetivado para o público.

3.2 Levantamento de Requisitos

A fim de prosseguir com a ferramenta, foram escaladas as funcionalidades necessárias para o *software* operar corretamente, localizando-as com base nas interações dos usuários com o sistema.

3.2.1 Requisitos Funcionais

São requisitos funcionais, as funções que interagem diretamente com o usuário, sendo catalogadas por meio da sigla RF, sua numeração de nome intuitivo ao seu comportamento.

RF01 – Cadastrar Perfil: Este comportamento indica a possibilidade de cadastrar usuários, no caso da Bloomy, o mentor cadastra o perfil;

RF02 – Realizar Login: Expressa a função do mentor conectar-se a sua conta na plataforma;

RF03 – Editar configurações: Permite que o mentor edite as configurações de experiência;

- RF04 – Editar perfil: Garante que o mentor edite suas informações;
- RF05 – Validar código: Possibilita, por meio de uma verificação ao código do mentor, a edição do perfil seja devidamente realizada;
- RF06 – Jogar fases: Proporciona ao usuário, em suma o estudante, usufruir dos jogos dispostos na plataforma e navegar entre as fases;
- RF07 – Verificar progresso: Expressa o progresso das fases, tornando visível aos usuários;
- RF08 – Realizar teste: Viabiliza o estudante a realizar o teste no fim de todos os jogos de uma fase;
- RF09 – Atualizar progresso: Contudo é possível atualizar o progresso conforme as atividades da plataforma são concluídas;
- RF10 – Ligar controle: Garante que o controle sensorial possa ser ligado;
- RF11 – Parear dispositivos: Proporciona a busca por dispositivos para conectar ao controle, pela rede Bluetooth;
- RF12 – Conectar com aparelho: Realiza a conexão do controle com o aparelho computador ou celular;
- RF13 – Enviar comandos de movimento: Permite a utilização dos botões do controle *bluetooth*, para enviar comandos de movimento para a plataforma e jogos;
- RF14 – Ativar vibração: Viabiliza a vibração do controle por meio de um conjunto de botões;
- RF15 – Desativar vibração: Garante que a vibração do controle seja desligada, por meio de um conjunto de botões;
- RF16 – Desligar controle: Expressa que o centro sensorial poderá ser desligado manualmente ou por baixa bateria.

3.2.2 Requisitos Não Funcionais

Ademais existem os requisitos não funcionais, representando as funções que não interagem diretamente com o usuário, sendo expressado pela sigla RNF, numeração e seu nome.

RNF01 – Intuitividade: Admite que a interface será simples de ser absorvida pelos estudantes, de maneira intuitiva e confortável;

RNF02 – Praticidade: Aponta que o sistema deve ser ágil na execução das funções e em sua navegabilidade;

RNF03 – Confortabilidade: A Bloomy proverá uma interface com cores confortáveis e relaxantes;

RNF04 – Sincronicidade: Os jogos e plataforma terão navegabilidade síncrona aos comandos do controle;

RNF05 – Personalizável: Permitirá personalizar experiência e preferências na plataforma;

RNF06 – Escalabilidade e Desempenho: O sistema será totalmente capaz de suportar os jogos existentes e os que serão dispostos na plataforma;

RNF07 – Suporte: A aplicação deverá dispor de informações na decorrência de dúvida sobre funcionalidades gerais;

RNF08 – *Feedback*: Deve fornecer o progresso no contínuo do estudante sobre seu desempenho nos estudos;

RNF09 – Usabilidade: Permitirá a navegabilidade na plataforma por meio do controle.

3.2.3 Regras de Negócios

Contudo, as regras de negócios definem normas a serem seguidas para o funcionamento geral do sistema, o que baseia as funcionalidades — as necessidades relacionadas com seu ambiente e seus usuários.

RN01 - Só é possível seguir as fases em ordem crescente e progressiva;

RN02 - É necessário completar uma unidade antes de ir para a próxima;

RN03 - É viável editar as configurações e perfil apenas se estiver na conta de mentores;

RN04 - Todos os manuais devem ser acessíveis, mesmo após o primeiro acesso a plataforma;

RN05 - É exequível ativar e desativar a vibração do controle;

RN06 - Todo mentor possui código parental, que garante acesso a funções restritas;

RN07 - Todas as funções da conta devem ser acessadas somente após a verificação do código parental.

3.3 Modelagem e Documentação do Sistema

A seguir, teremos a representação do software a modelagem das funcionalidades nos diagramas de UML, para melhor visualização, o que auxilia no progresso geral da Bloomy.

3.3.1 Diagrama de Casos de Uso

Com ele é possível representar de forma sucinta, o funcionamento da plataforma, de maneira geral com as interações pertinentes do mentor e estudante com os casos de uso.

<Diagrama>

Como representado acima, é possível ver tudo que os atores mentor e estudante podem fazer. Sendo possível, em suma, o mentor cadastrar e editar o perfil, que será usado para ambos os usuários — apenas o mentor possui acesso para realizar tais procedimentos — podendo também ver o progresso realizado nas atividades ao fazer login. O estudante por sua vez apenas usufrui dos jogos, avançando até realizar o teste de cada fase.

3.3.2 Documentação do Caso de Uso

Ao documentar o diagrama de casos, é possível entender e visualizar os detalhes implícitos em cada funcionalidade, tornando o sucinto em específico, para uma análise mais profunda sobre o protótipo em si.

3.3.3 Diagrama de Atividades

Para expressar as situações prováveis do sistema, é utilizado este diagrama, frisando as etapas para a completude da atividade. Separando as atividades em cada ator que a realiza, e o fluxo de ações até o fim de cada situação.

<Atividade estudante>

Acima é notado o fluxo principal do estudante, partindo da navegação até acesso aos jogos de cada unidade e matéria, até responder o teste final.

<Atividade mentor editar>

A seguir temos a primeira atividade possível para o mentor, onde o cadastro e login são realizados, focando na possível edição do perfil apenas com o uso do código parental, não permitindo a edição caso a verificação seja falsa.

<Atividade mentor personalizar>

Contudo, a outra situação importante para o mentor são as etapas de personalizar a experiência, possibilitando esta ação apenas com o uso do código, não sendo possível caso condição não seja verdadeira.

3.3.4 Diagrama de Sequência

É desejável entender e expressar as mensagens e objetos no fluxo de sequência, sendo este diagrama o ideal, localizando decisões e divergências durante a interação sequenciada.

3.3.5 Diagrama de Máquina de Estados

Esta seção destaca a mudança dos estados, para permitir um funcionamento modular e entendimento das funcionalidades da Bloomy.

Em geral, é demonstrado a mudança entre o estado de entrar ao sistema e passar sobre a verificação do login, possibilita a escolha entre o fluxo de estados do estudante, onde se joga e realiza o teste final, e o do mentor, onde a edição do perfil e personalização da experiência com o uso da verificação de acesso.

3.4 Estrutura da Plataforma Digital

A Bloomy foi planejada para ser uma aplicação multiplataforma — sendo disponível para computadores e dispositivos móveis — exemplificados em wireframes de baixa e alta fidelidade, com cores, ícones e disponibilidade de elementos específicas.

Foram implementados na plataforma tons de roxo remetendo a ludicidade dos jogos, com o laranja que representa o entusiasmo, que nos tons certos se equilibram com um contraste menos agressivo aos olhos — o destaque da cor quente é amenizado com a frieza da cor oposta — quando combinadas mesmo tais cores complementares caracterizam a identidade do sistema.

O *software* também foi disposto com a interface gráfica, no modelo de paisagem — tela em posição horizontal — evitando possíveis desconfortos por transição de

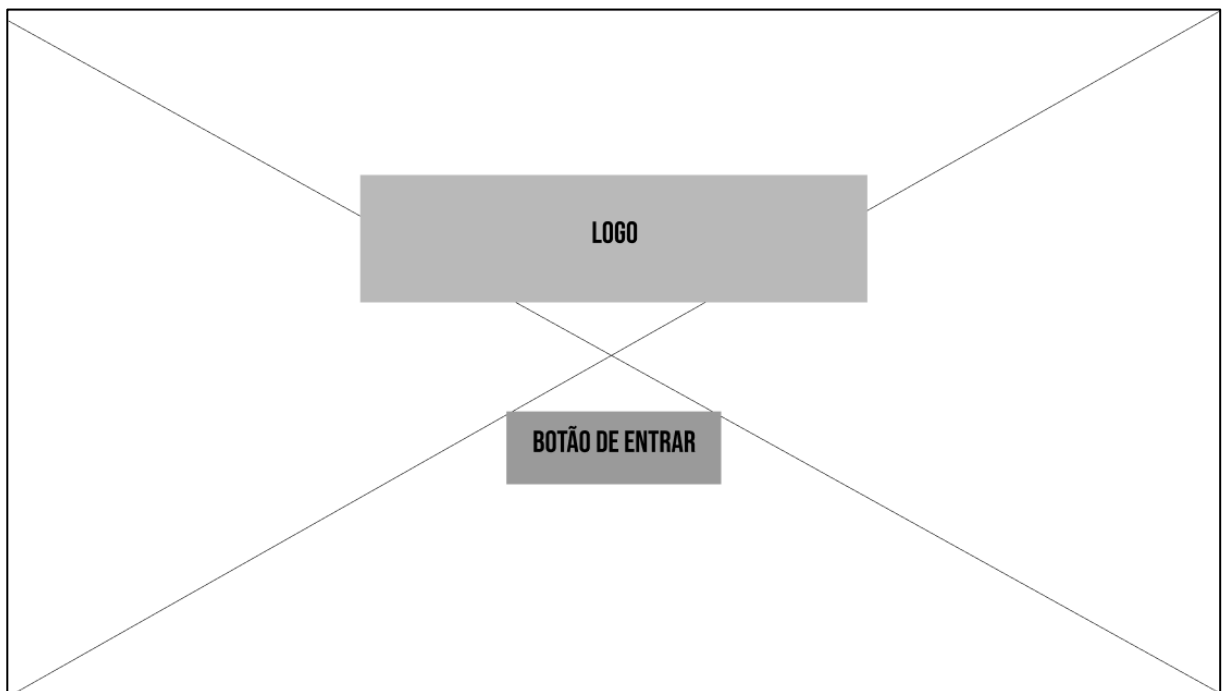
modelos, padronizando a experiência da plataforma com os jogos em dispositivos móveis e computadores.

3.5 Wireframes de Baixa e Alta Fidelidade

Foram montados para exemplificar as páginas e navegação da Bloomy, utilizando a plataforma de design Figma para o desenvolvimento dos *wireframes*, demonstrando a interface com zelo a sua versão final.

O primeiro *wireframe* representa a tela de início, como uma breve introdução ao sistema com uma imagem promocional e um botão de “Entrar” para navegar ao login:

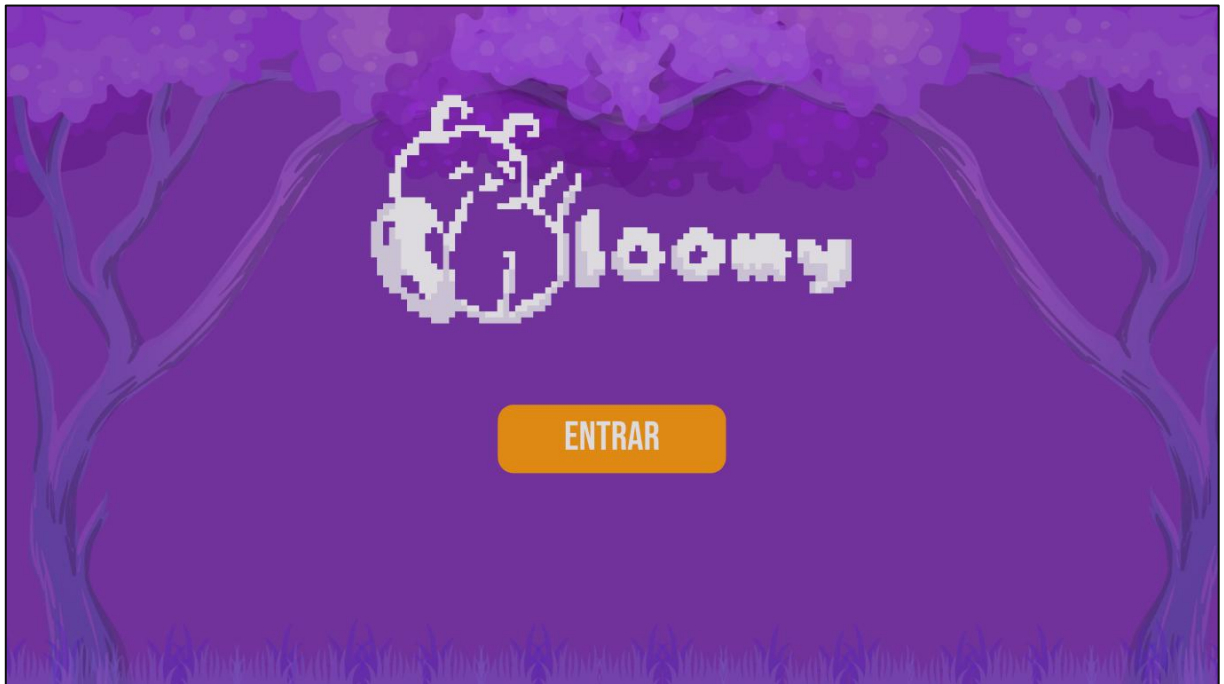
Figura 27 — Wireframe de Baixa Fidelidade 01



Fonte: Autoria Própria (2025).

E assim, ficou o *wireframe* de alta fidelidade:

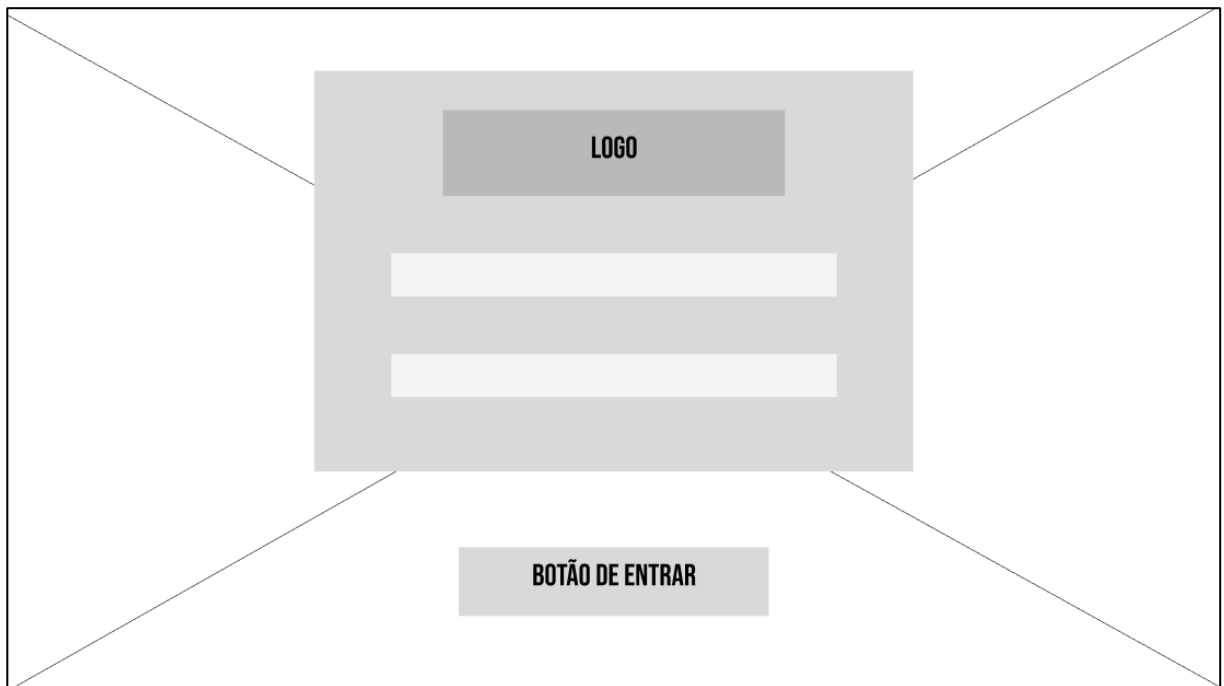
Figura 28 — Wireframe de Alta Fidelidade 02



Fonte: Autoria Própria (2025).

No segundo, possui um formulário simples de login, com um link para o cadastro caso o usuário não possua conta:

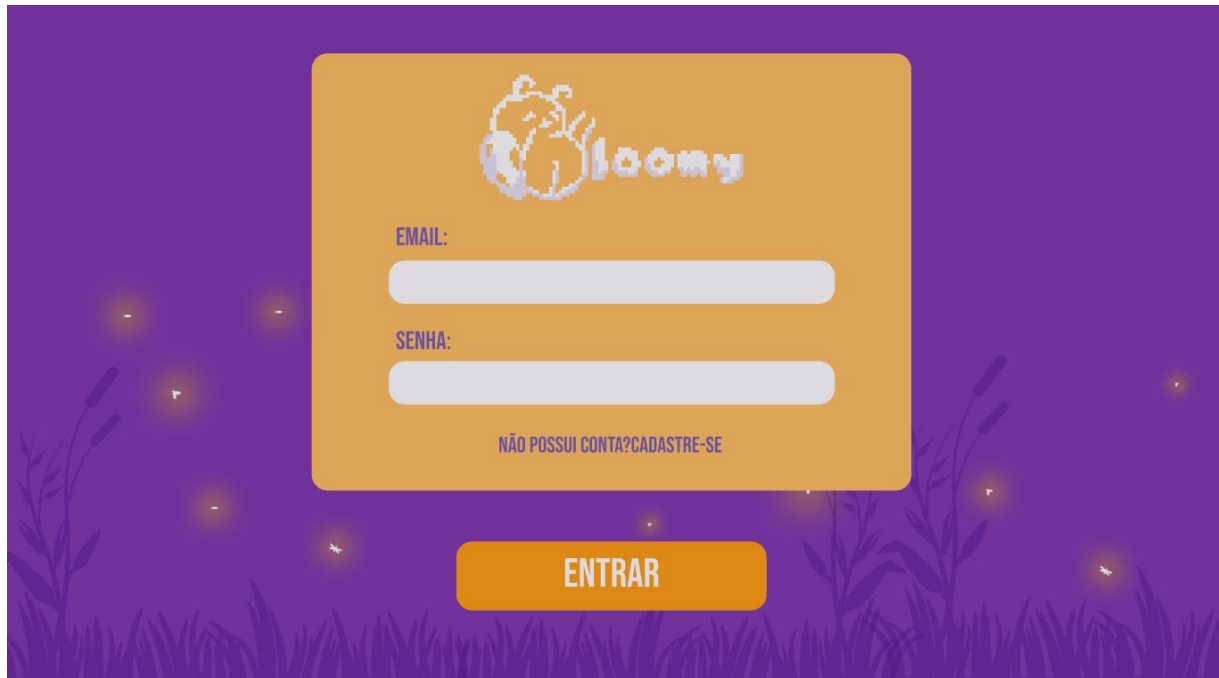
Figura 29 — Wireframe de Baixa Fidelidade 02



Fonte: Autoria Própria (2025).

Abaixo temos a versão finalizada dessa tela, onde foram aplicados os componentes finalizados e as devidas cores:

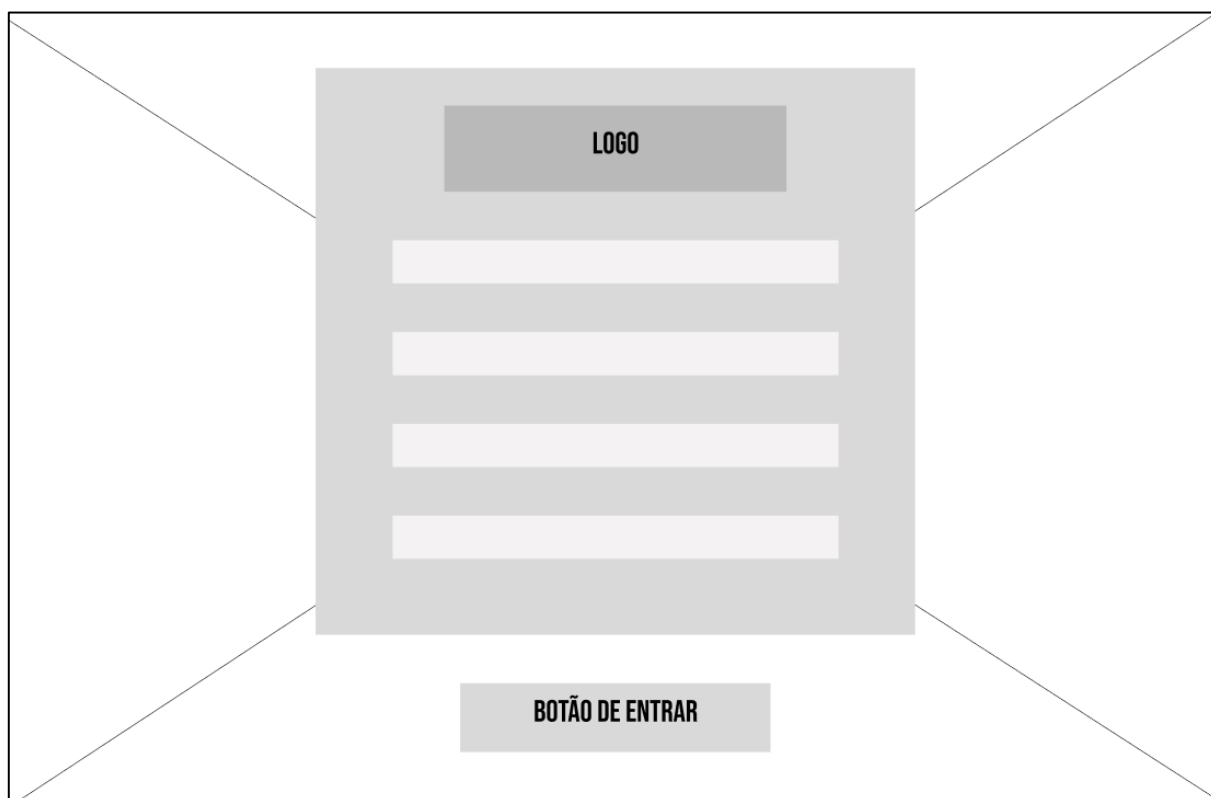
Figura 30 — Wireframe de Alta Fidelidade 02



Fonte: Autoria Própria (2025).

Abaixo temos a tela de cadastro, onde é possível criar uma conta informando o nome do estudante, do responsável, email e senha:

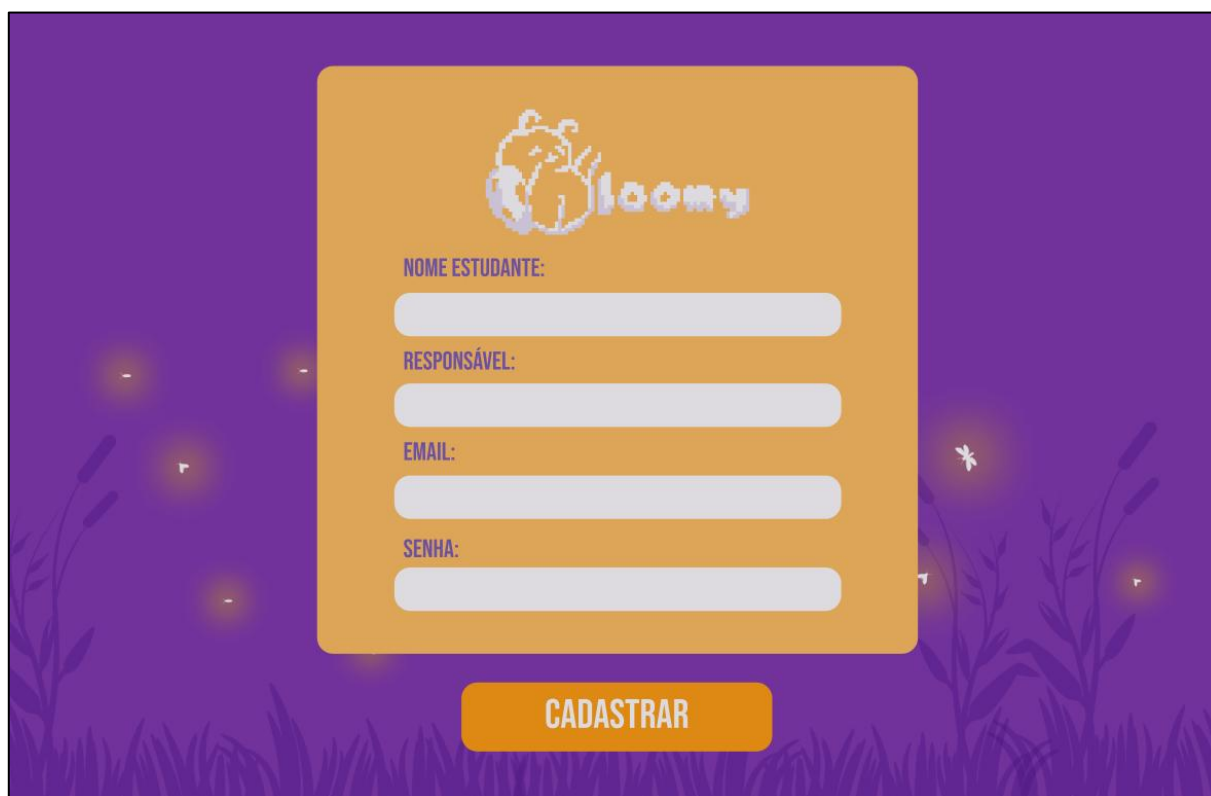
Figura 31 — Wireframe de Baixa Fidelidade 03



Fonte: Autoria Própria (2025).

Com a implementações das cores, textos e ícones, é visível abaixo a versão final do formulário:

Figura 32 — Wireframe de Alta Fidelidade 03

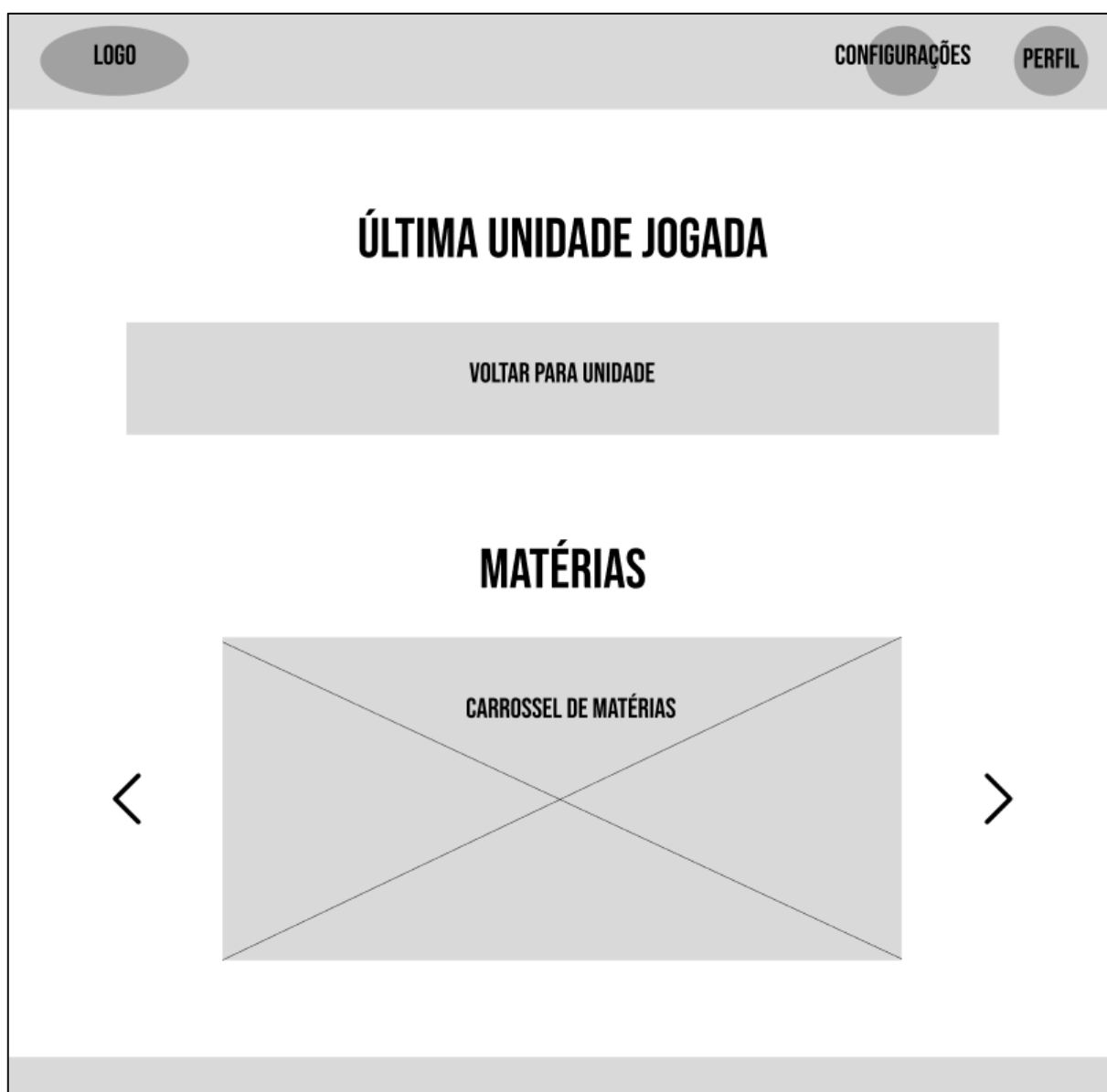


The wireframe shows a registration form centered on a purple background with a night garden theme. The form is contained within a light orange rounded rectangle. At the top of this rectangle is the 'loomy' logo, which features a stylized animal head. Below the logo are four input fields, each preceded by a label in all caps: 'NOME ESTUDANTE:', 'RESPONSÁVEL:', 'EMAIL:', and 'SENHA:'. Each label is followed by a light gray rounded rectangular input box. At the bottom of the orange rectangle is an orange rounded button with the text 'CADASTRAR' in white, uppercase letters. The background of the entire page is purple and decorated with faint silhouettes of plants and glowing light spots.

Fonte: Autoria Própria (2025).

A seguir, temos a tela de menu, que permite a navegação pela Bloomy, com as opções de selecionar matéria e retornar a última unidade visitada:

Figura 33 — Wireframe de Baixa Fidelidade 04



Fonte: Autoria Própria (2025).

Contudo, o menu com a aplicação do design fica desta maneira:

Figura 34 — Wireframe de Alta Fidelidade 04



Fonte: Autoria Própria (2025).

Pensando nas funcionalidades, foi criado as configurações, onde é possível editar da experiência no aplicativo:

Figura 35 — Wireframe de Baixa Fidelidade 05



Fonte: Autoria Própria (2025).

Na versão atual, as configurações permitem que o mentor escolha se quer trocar o código de acesso, receber notificações da Bloomy e quando quer receber — ao ativar o modo Pomodoro, as notificações vão de 25 em 25 minutos.

Figura 36 — Wireframe de Alta Fidelidade 05



Fonte: Autoria Própria (2025).

Nesta representação, é exemplificada a tela de perfil, onde o mentor pode checar as informações cadastradas.

Figura 37 — Wireframe de Baixa Fidelidade 06



Fonte: Autoria Própria (2025).

Agora com as implementações finais, é possível editar as informações cadastradas após a verificação do Código de acesso — para que apenas o mentor que possui a senha possa editar.

Figura 38 — Wireframe de Alta Fidelidade 06

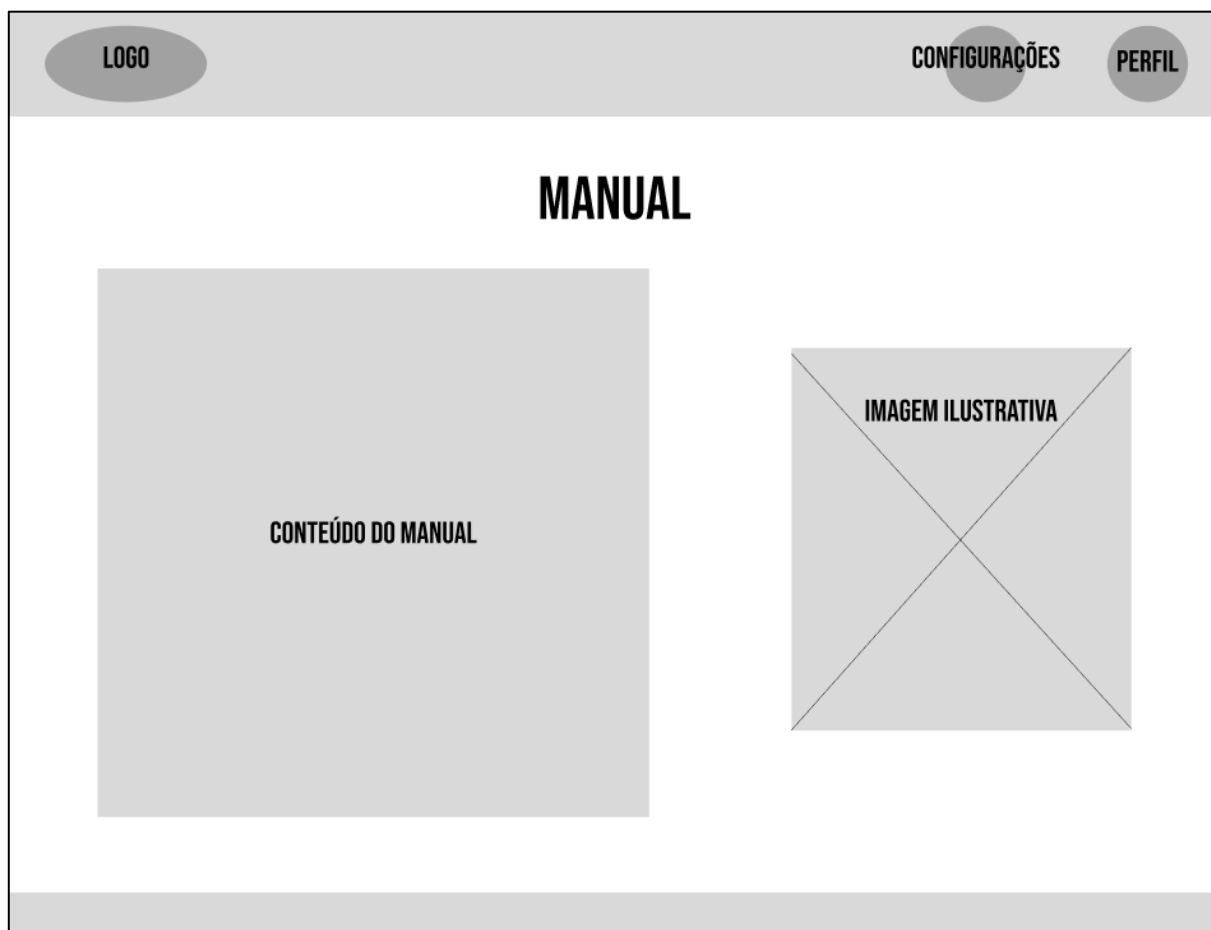


O wireframe apresenta a interface de perfil de usuário. No topo, uma barra de cabeçalho roxa contém o logo 'loomy' à esquerda e ícones de engrenagem e perfil de usuário à direita. O título 'PERFIL' está centralizado na seção principal. Abaixo dele, um cartão amarelo contém um círculo de perfil com o texto 'CÓDIGO' abaixo. À direita do cartão, há quatro campos de texto empilhados: 'MENTOR', 'ESTUDANTE', 'EMAIL@EMAIL.COM' e 'SENHA'. Um botão laranja 'EDITAR' está posicionado na base do cartão.

Fonte: Autoria Própria (2025).

Com um sistema baseado em um controle para navegação, foi criada a página de manuais, que explicam o funcionamento da IoT, como pareá-la, e movimentar-se por ela.

Figura 39 — Wireframe de Baixa Fidelidade 07



Fonte: Autoria Própria (2025).

Na versão atual, é possível perceber o manual de funcionamento do controle, com a adição de textos e imagens para tornar a experiência mais intuitiva.

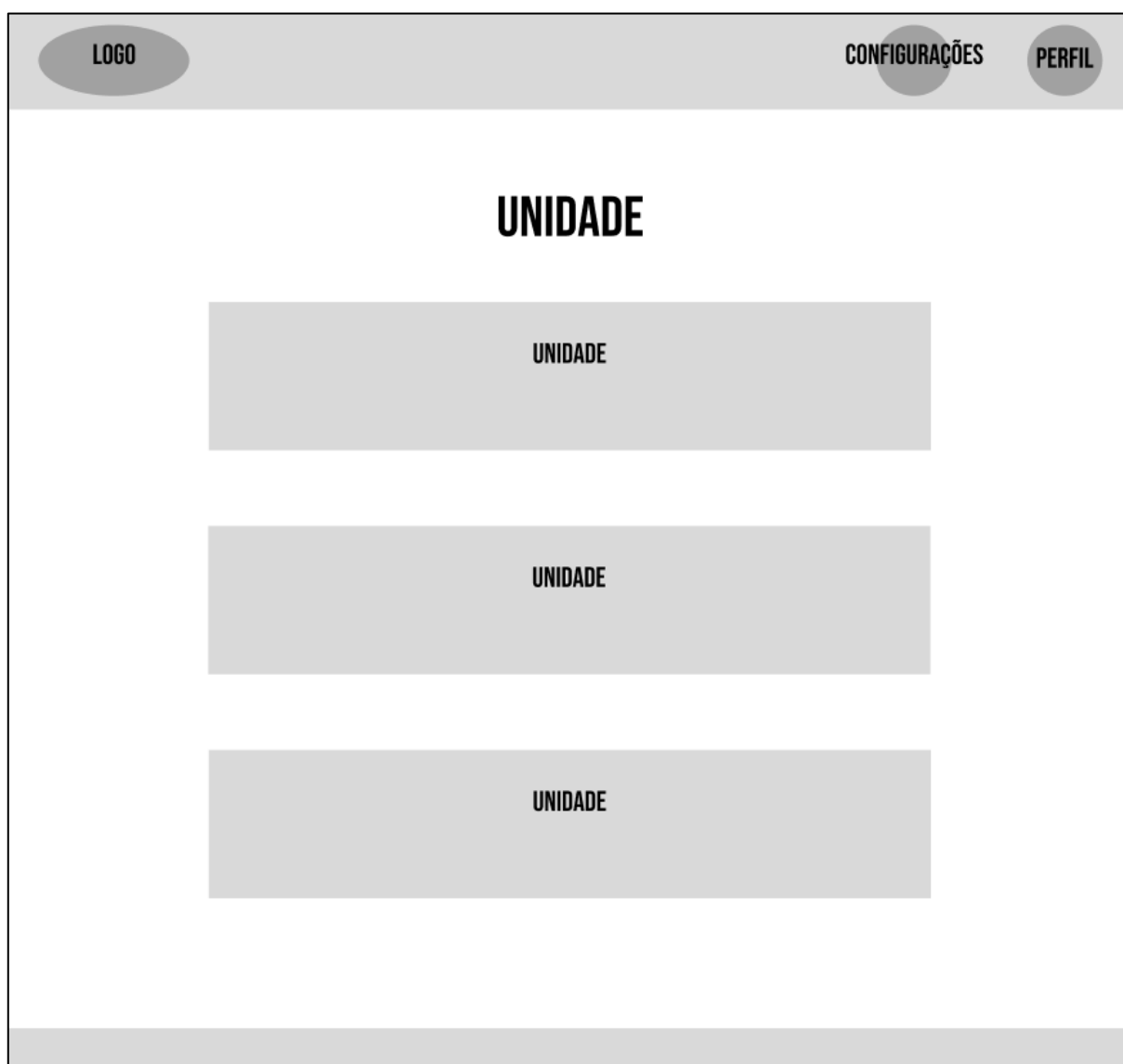
Figura 40 — Wireframe de Alta Fidelidade 07



Fonte: Autoria Própria (2025).

O próximo *wireframe*, mostra a tela de unidades que possuem os níveis de cada jogo, sendo a unidade um a mais simples e a unidade quatro a mais complexa.

Figura 41 — Wireframe de Baixa Fidelidade 08



Fonte: Autoria Própria (2025).

Consequente, a tela de unidades é finalizada com a aplicação das cores e tipografia, também com o progresso realizado em cada matéria.

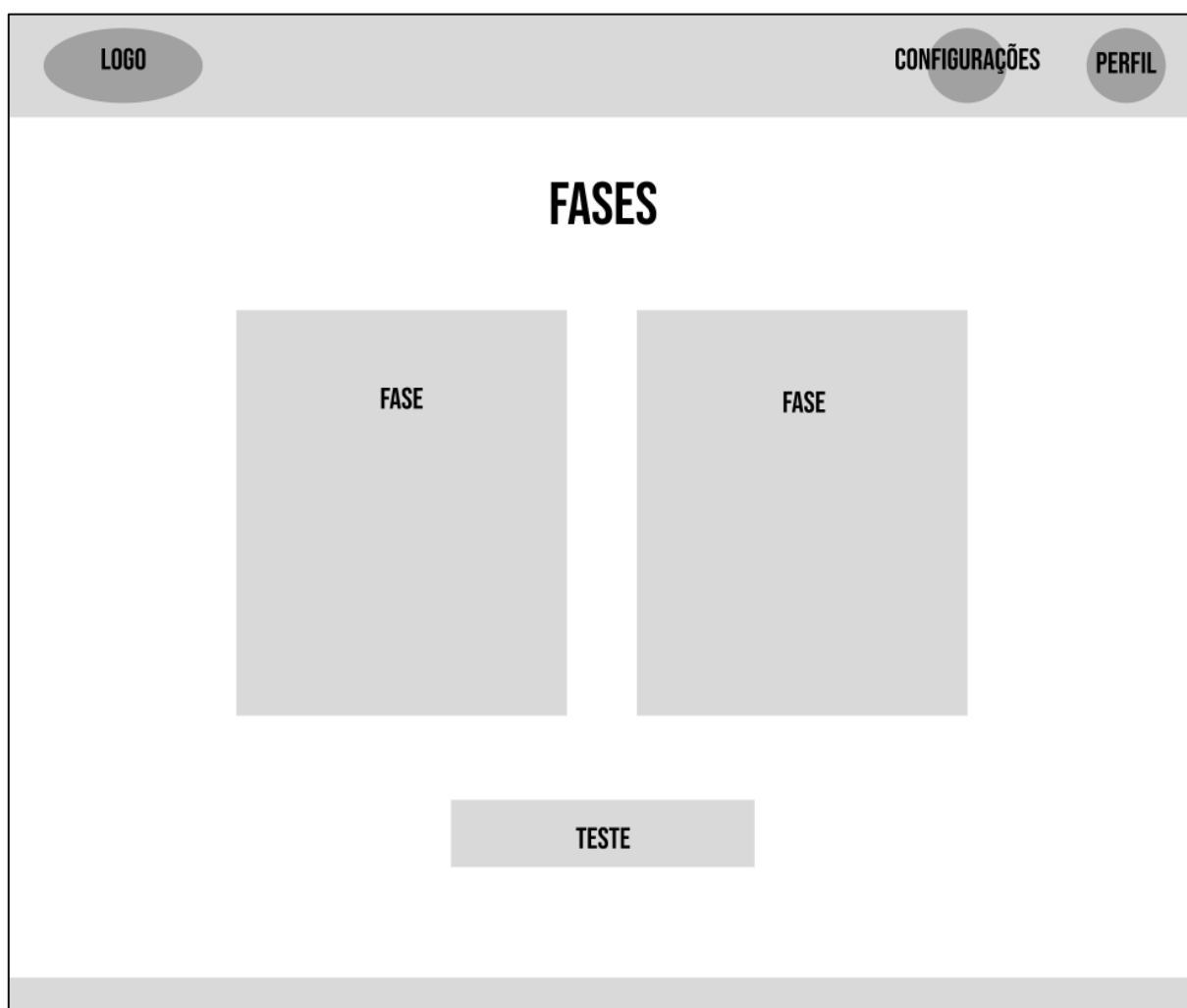
Figura 42 — Wireframe de Alta Fidelidade 08



Fonte: Autoria Própria (2025).

A seguir, temos a página que se encontram os jogos da unidade, sendo são dois jogos por unidade com o botão que redireciona para o teste final.

Figura 43 — Wireframe de Baixa Fidelidade 09



Fonte: Autoria Própria (2025).

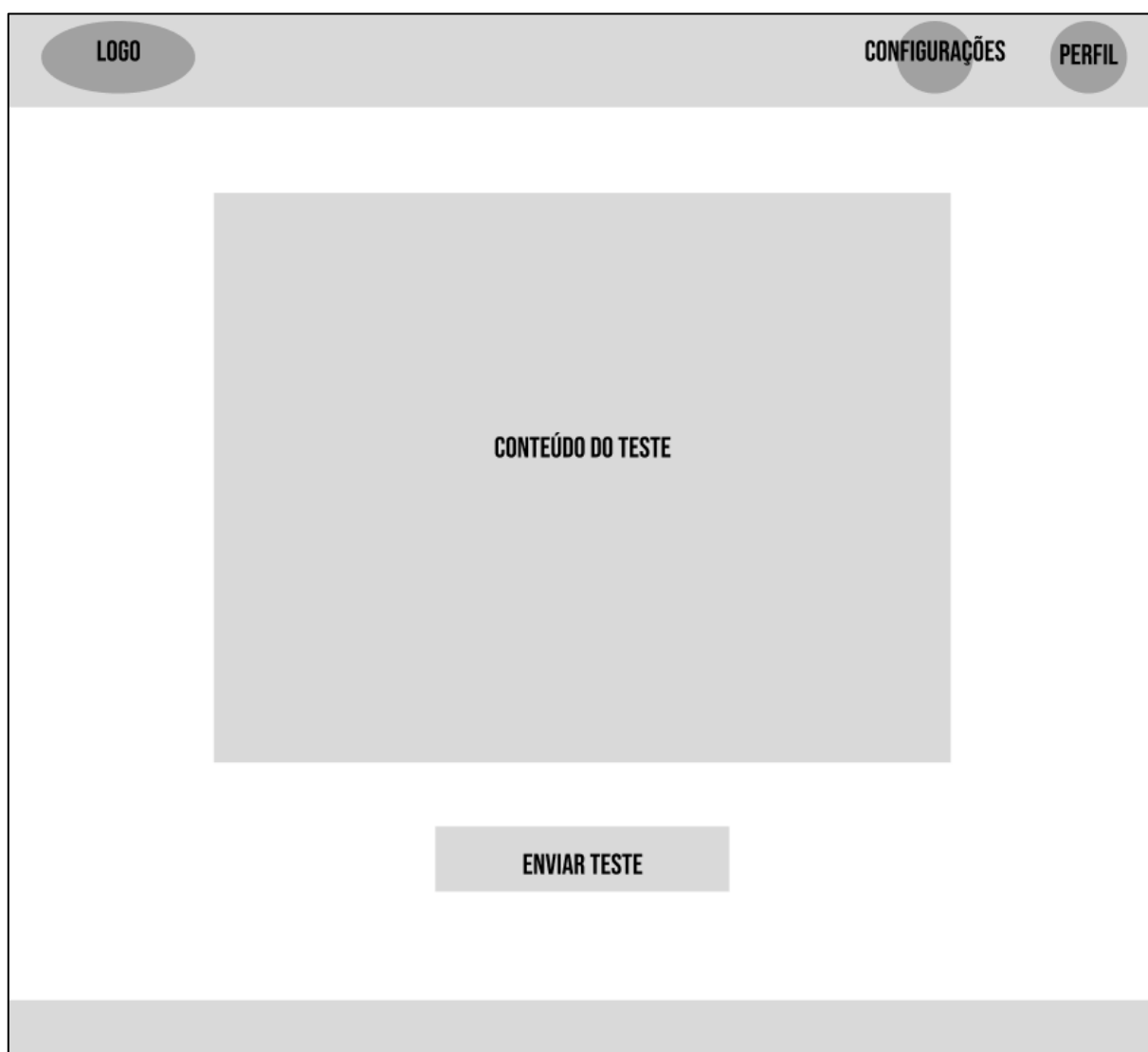
Ao finalizar, foi adicionado os ícones para a fase desbloqueada, fase bloqueada e seus nomes.

Figura 44 — Wireframe de Alta Fidelidade 09



Fonte: Autoria Própria (2025).

Figura 45 — Wireframe de Baixa Fidelidade 10



Fonte: Autoria Própria (2025).

Figura 46 — Wireframe de Alta Fidelidade 10



The wireframe shows a web interface for a test. At the top, there is a purple header bar. On the left of the header is a logo with a cartoon character and the text 'loomy'. On the right are two circular icons: a gear for settings and a person for a profile. Below the header, the word 'TESTE' is centered in a large, bold, purple font. The main content area has a light purple background. In the center, there is a light orange rounded rectangle containing two identical question blocks. Each block starts with the text 'PERGUNTA 1 - QUAL O NÚMERO QUE FICA ENTRE 5 E 7?' in purple. Below each question are three radio button options: 'A) 6', 'B) 8', and 'C) 9'. At the bottom of the orange box, there is a purple button with the text 'ENVIAR TESTE' in white.

Fonte: Autoria Própria (2025).

3.6 Desenvolvimento dos Jogos

Foram desenvolvidos no total oito jogos, utilizando a plataforma de Construct 3 para programar a mecânica e elaborar cada layout, resultando em dois jogos para cada matéria.

Contudo as matérias selecionadas foram: português, matemática, inglês e ciências — que estão dispostas na base escolar do Fundamental I — com dinâmicas simples, baseadas nos clássicos de fliperama como *Space Invaders* e *Frogger*, também em regras isoladas de jogos como *Omori*, *CatJumper* e *Stardew Valley*.

Para cativar o público selecionado, é necessário criar personagens envolventes que se conectem com o jogador. Por isso, nasceram Luke o cágado, Ruby a joaninha,

Pipo o morcego e Vulpy a raposa, cada um para uma matéria específica, sendo respectivamente, Ciências, Português, Matemática e Inglês.

Figura 47 — Sprites dos Personagens



Fonte: Autoria Própria (2025).

3.6.1 Mecânicas Aplicadas

Os jogos foram desenvolvidos com a mecânica que melhor ornasse com o objetivo escolhido para cada situação, sendo divertido de jogar e ainda se mantém simples de replicar.

A matéria de matemática é protagonizada por Vulpy a raposa, onde no M1 — um labirinto no supermercado — ela precisa fazer a compra de todos os itens na lista, resolvendo *puzzles* de lógicos para conseguir sair com a lista completa. Já no M2, é preciso utilizar novamente da lógica para alcançar o número 15, somando e subtraindo valores com caixas de números positivos e negativos.

Nos jogos de ciências, o jogador controla Luke o cágado, o C1 simula o plantio de um girassol, desde semear, regar a colher, também sendo necessário defender a flor das pragas — os ratos — que aparecem na plantação para destruí-la. Em C2, é necessário coletar o material reciclável das ruas e colocar em suas respectivas lixeiras, tomando cuidado para não ser atropelado pelos carros das avenidas

Para a língua portuguesa, Ruby a joaninha, foi escolhida como protagonista, no P1 é necessário coletar a palavra escrita de maneira correta, enquanto percorre por plataformas assegurando-se de que não irá cair dos prédios da cidade. P2 é um

grande jardim, e o jogador precisa levar as letras na ordem certa para completar a frase disposta no meio do mapa.

Contudo, na língua inglesa, I1 é explorada por Pipo o morcego, que precisa acertar projéteis nas nuvens que se relacionam com o tema da nuvem central. Por fim, I2 onde é necessário coletar todos os verbos em ingles dispostos no meio de palavras quaisquer.

3.6.2 Conceitos Visuais

Como visto na seção anterior, para cada matéria existe um personagem diferente, para uma conexão maior com os jogadores, foi necessário criar quadros de animação com esses *Sprites*, para tornar a experiência flúida e divertida.

Cada quadro de animação foi desenvolvido com base nos comportamentos que cada mascote possui em seus ambientes, direções podem se movimentar: cima, baixo, esquerda e direita.

A implementação dos cenários foi baseada no tipo de aventura que cada personagem iria enfrentar, elaboradas com o que mais se encaixava com o objetivo e mecânica dos jogos. Desde supermercados, salas mágicas,

3.7 Construção do Controle Interativo

3.7.1 Circuito e Funcionamento

3.7.2 Modelagem e Impressão

4 CONSIDERAÇÕES FINAIS

Em suma, ludificar os exercícios passados em sala — por meio de jogos — possibilita o entendimento e pratica até mesmo fora da sala de aula, o que torna um meio viável de fortalecer a educação e auxiliar os alunos que se sintsm desconectados da rede de ensino.

REFERÊNCIAS

ACETI, L. B.; BRITO, L. T.; OLIVEIRA, D. C. M.; PINA, D. B.; KUNSTMANN, L. N. O. **Desenvolvimento de uma Aplicação Web para Instrumentação de Scripts para a Ferramenta DfAnalyzer**. 2023. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Universidade Federal Fluminense, Niterói, 2023.

ALURA. 2024. **O que é Firebase? Para que serve, principais característica e um guia dessa ferramenta Google**. Disponível em: <https://www.alura.com.br/artigos/firebase?srsItid=AfmBOopem7O3TvcbgX7pGKqJmj6aesJI7WIJtDFoEtQfEpk0PCVOwqyC>. Acesso em: 28 abr. 2025.

ALVES, L. F. D.; AZEVEDO, V. N. DE L.; COSTA JUNIOR, A. DE O. Construct 2: Oficinas Formativas Para a Criação de Jogos Educacionais Digitais. **III Simpósio Ibero-Americano de Tecnologias Educacionais**, jun. 2019.

ANACLETO CHICCA JUNIOR, N.; GOMEZ CASTILLO, L. **Os desafios em utilizar a impressão 3D no processo ensino-aprendizagem de design**. RENOTE, Porto Alegre, v. 16, n. 1, 2018. DOI: 10.22456/1679-1916.86040. Disponível em: <https://seer.ufrgs.br/index.php/renote/article/view/86040>. Acesso em: 29 abr. 2025.

APOCALYPSE, Marcos; JORENTE, Maria. **O MÉTODO DESIGN THINKING E A PESQUISA EM CIÊNCIA DA INFORMAÇÃO**. Encontros Bibli, Florianópolis, v.27, n.1, p. 1-21, nov. 2022.

ARGOUD, P. **Modelagem 3D**. São Paulo: SENAC, 2024. Disponível em: <https://play.google.com/books/reader?id=iCgGEQAAQBAJ&pg=GBS.PP1&hl=ptBR>. Acesso em: 29 abr. 2025. E-book.

ATENA EDITORA (Org.). **Metodologia da pesquisa em educação e ensino de ciências**. Ponta Grossa: Atena Editora, 2023.

AZZOLINI, M. G. **Derivação de um estilo arquitetural para o front-end de sistemas baseados em React.js com base em projetos Open Source**. Ufrgs.br, 2021.

BEZERRA, Franklyn Seabra Rogério. **Desenvolvimento Nativo vs Ionic vs React Native: uma análise comparativa do suporte à acessibilidade em Android**. 2021. 25 f. Trabalho de Conclusão de Curso (Graduação em Computação)-Universidade Federal do Ceará, Fortaleza, 2021.

BERTOLETI, P. **Projetos com ESP32 e LoRa**. São Paulo: INSTITUTO NEWTON C. BRAGA, 2019.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.

CARVALHO, L.; ARANHA, G.; CARDIA NETO, J. B. . JOGOS DIGITAIS. **Interface Tecnológica**, v.18, n.2, p. 264-267, 14 maio, 2022.

DATE, C. J. Tradução de Daniel Vieira. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Elsevier Brasil, 2004.

ELMASRI, Ramez; NAVATHE, Shamkant B. **SISTEMAS DE BANCO DE DADOS**. 6. ed. São Paulo: Addison Wesley, 2011.

ESCUDELARIO, B.; PINHO, D. **React Native**. São Paulo, SP: Casa do Código, 2021.

ESTADO DE MINAS. **Uso excessivo de telas pode causar demência digital, diz neurocirurgião**. 6 out. 2023. Disponível em: https://www.em.com.br/app/noticia/saudeebemviver/2023/10/06/interna_bem_viver,1_572765/uso-excessivo-de-telas-pode-causar-demencia-digital-diz-neurocirurgiao.shtml . Acesso em: 19 abr. 2025.

FEDERAL DE SANTA, U. et al. Revista eletrônica de biblioteconomia e ciência da informação: O MÉTODO DESIGN THINKING E A PESQUISA EM CIÊNCIA DA INFORMAÇÃO Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação. v. 27, p. 1–21, 2022.

FLANAGAN, D. Tradução de João Eduardo Nóbrega Tortello. **JavaScript: o Guia Definitivo**. 6. ed. Porto Alegre: Bookman, 2013.

FLATSCHART, Fábio. **HTML 5: Embarque Imediato**. Rio de Janeiro, RJ: Brasport, 2011.

FOWLER, Martin. **UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos**. 3. ed. Porto Alegre: Bookman, 2005.

FREIRE, Paulo. **Pedagogia da autonomia: saberes necessários à prática educativa**. São Paulo: Paz e Terra, 1996.

FROBOESE DA SILVA, Eloar; MENIN DA SILVA, Lucas; GONÇALVES DEON, Vinícius; TOSO, Marcelo André. IMPRESSÃO 3D APLICADA À TECNOLOGIA ASSISTIVA. **Destaques Acadêmicos**, [S. l.], v. 12, n. 4, 2020. DOI: 10.22410/issn.2176-3070.v12i4a2020.2657

GUEDES, Gilleanes T. A. **UML 2: uma abordagem prática**. 3.ed. São Paulo: Novatec, 2009.

G1. '**Cérebro podre**': entenda o que é 'brain rot', expressão do ano eleita pelo Dicionário Oxford. G1 Educação, 02 dez. 2024. Disponível em: <https://g1.globo.com/educacao/noticia/2024/12/02/entenda-o-que-e-brain-rot-cerebro-podre-expressao-do-ano-eleita-pelo-dicionario-oxford.ghtml>. Acesso em: 12 maio 2025.

HORSTMANN, Cay. **Conceitos de computação com o essencial de C++**. Tradução de Carlos Arthur Lang Lisbôa e Mária Lúcia Blanck Lisbôa. 3ªed. Porto Alegre: ARTMED, p. 30- 2008. Disponível em: https://play.google.com/store/books/detailsid=qSn8wEMsolMC&rdid=bookhttps://play.google.com/store/books/detailsid=qSn8wEMsolMC&rdid=bookqSn8wEMsolMC&rdot=1&source=gbs_vpt_read&pcampaignid=books_booksearch_viewportqSn8wEMsolMC&rdot=1&source=gbs_vpt_read&pcampaignid=books_booksearch_viewport. Acesso em 26 Abr. 2025. E-book.

JOBSTRAIBIZER, Flávia. **Criação de Sites CSS**. São Paulo, SP: Digerati Books, 2009.

LARMAN, Craig. **Utilizando UML e padrões**: uma introdução à análise e ao projeto orientados a objeto. Porto Alegre: Bookman, 2000.

MACHADO, Kheronn Khennedy. **Angular 11 e Firebase**: construindo uma aplicação integrada com a plataforma do google. São Paulo: Casa do Código, 2021. 177 p.

MAGRANI, E. **A internet das coisas**. Rio de Janeiro: FGV Editora, 2018. 192 p.

MANOVICH, L. Banco de Dados. **Eco-Pós**, [S. l.], v. 18, n. 1, p. 7–26, 2015. Disponível em: https://ecopos.emnuvens.com.br/eco_pos/article/view/2366. Acesso em: 28 abr. 2025.

MARQUES, Jideon. **O Curso de Codificação C++**: Santa Catarina: Clube de Autores, 2023. 658 p. Disponível em: https://www.google.com.br/books/edition/O_Curso_De_Codificação_C++/QJbqEAAAQBAJ?hl=pt-BR. Acesso em: 01 maio 2024. E-book.

MARQUES, P. B.; CASTANHO, M. I. S. O que é a escola a partir do sentido construído por alunos. **Psicologia Escolar e Educacional**, v. 15, n. 1, jun. 2011.

MARTINS, Mateus Rodrigues. **Desenvolvimento de uma biblioteca em C++ para modelagem e controle cinemático de robôs manipuladores usando quatérnios duais**. 2013. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) — Universidade Federal de Minas Gerais, Belo Horizonte, 2013.

MILETTO, E. M.; DE CASTRO BERTAGNOLLI, S. **Desenvolvimento de Software II: introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP - eixo: informação e comunicação - Série Tekne**. Porto Alegre: Bookman Editora, 2014.

MIRANDA MORANDINI, M.; DEL VECHIO, G. H. IMPRESSÃO 3D, TIPOS E POSSIBILIDADES: uma revisão de suas características, processos, usos e tendências. **Interface Tecnológica**, Taquaritinga, SP, v. 17, n. 2, p. 67– 77, 2020. DOI: 10.31510/infa.v17i2.866.

MORAIS, J. V. S. **ESP32 com IDF**. São Paulo: Editora NCB, 2023.

MORORÓ, J. F. **Um estudo comparativo entre JavaScript e TypeScript**. 2024. 154 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) – Campus de Sobral, Universidade Federal do Ceará, Sobral, 2024. Acesso em: 28 de Abril, 2025.

NASCIMENTO, A.; SOUSA, C.; OLIVEIRA, G.; OLIVEIRA, F.; CAMPOS, G. **Controle de Iluminação Através da Internet Utilizando as Tecnologias ReactJS, Firebase e ESP32**. Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574), [S. l.], v. 1, n. 6, p. 46–49, 2023. Disponível em: <https://periodicos.ufersa.edu.br/ecop/article/view/11835>. Acesso em: 26 abr. 2025.

NEVES.V. React: o que é, como funciona e um Guia dessa popular ferramenta JS. Alura (2023).

PAGANI, T. **Design thinking**. [s.l.] SENAC, p. 3–9, 2018. Disponível em: [https://play.google.com/store/books/details?id=lz1MDwAAQBAJ&rdid=bookhttps://play.google.com/store/books/details?id=lz1MDwAAQBAJ&rdid=book-lz1MDwAAQBAJ&rdot=1&source=gbs_vpt_read&pcampaignid=books_booksearch_viewportlz1MDwAAQBAJ&rdot=1&source=gbs_vpt_read&pcampaignid=books_books_e_arch_viewport](https://play.google.com/store/books/details?id=lz1MDwAAQBAJ&rdid=book-https://play.google.com/store/books/details?id=lz1MDwAAQBAJ&rdid=book-lz1MDwAAQBAJ&rdot=1&source=gbs_vpt_read&pcampaignid=books_booksearch_viewportlz1MDwAAQBAJ&rdot=1&source=gbs_vpt_read&pcampaignid=books_books_e_arch_viewport) . Acesso em: 17 maio. 2025. E-book.

PAULA, B. H. de. **Jogos digitais como artefatos pedagógicos**: o desenvolvimento de jogos digitais como estratégia educacional. 2015. Dissertação (Mestrado em Artes Visuais) - Instituto de Artes, Universidade Estadual de Campinas, Campinas, 2015.

PIMENTEL, Pedro Paulo Cavalcante. **Gamificação na educação matemática**: desenvolvimento de jogos 2D no Construct 3 para engajar nativos digitais. 2025. Dissertação (Mestrado em Matemática) – Centro de Ciências Exatas e da Natureza, Universidade Federal da Paraíba, João Pessoa, 2025.

PUREWAL, S. Tradução de Lúcia Kinoshita. **Aprendendo a desenvolver aplicações web**: Desenvolva rapidamente com as tecnologias JavaScript mais modernas. São Paulo: Novatec Editora, 2014.

RODRIGUES, Amanda Anjolin. **Desenvolvimento de aplicativo Android utilizando os serviços de firebase**, 2021. Trabalho de conclusão de curso (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) Faculdade de Tecnologia de São Paulo, São Paulo, SP, 2021.

RODRIGUES, D. **APRENDA C++ MODERNO**: Desenvolva Aplicações Performáticas com Recursos Avançados. Dos Fundamentos às Aplicações Práticas. [s.l.] StudioD21, 2024.

SCHEIDT, F. A. **Fundamentos de CSS**: criando design para sistemas web. Foz do Iguaçu: [s.n.], 2015.

SENA, S. de; SCHMIEGELOW, S. S.; PRADO, G. M. B. C. do; PERASSI, R.; FIALHO, F. A. P. **Aprendizagem baseada em jogos digitais**: a contribuição dos jogos epistêmicos na geração de novos conhecimentos. CINTED-UFRGS Novas Tecnologias na Educação, v. 14, n. 1, jul. 2016.

SILVA, Camila da. **Abandono escolar atinge recorde histórico entre crianças e adolescentes do Ensino Fundamenta, mostra IBGE**. Carta Capital, 22 mar. 2024. Disponível em: <https://www.cartacapital.com.br/educacao/abandono-escolar-atinge-recorde-historico-entre-criancas-e-adolescentes-do-ensino-fundamental-mostra-ibge/>. Acesso em: 28 out. 2025.

SILVA, Denys Alves; DE SOUZA, Caio Frias. Construção de App com React Native. **TECNOLOGIAS EM PROJEÇÃO**, [S. l.], v. 10, n. 1, p. 1–15, 2019.

SILVA, Eduardo Corneto; ESPEJO, Márcia M. S. B. Adoção da Internet das Coisas (IoT) na agropecuária: uma revisão sistemática sobre as possibilidades de adoção no ambiente produtivo rural brasileiro. **Interações**, Campo Grande, v. 25, n. 4, 2024.

SILVA, Maurício Samy. **HTML 5: A LINGUAGEM DE MARCACAÇÃO QUE REVOLUCIONOU A WEB**. São Paulo, SP: Novatec Editora, 2011.

SILVA, Maurício Samy. JavaScript - **Guia do Programador**: Guia completo das funcionalidades de Linguagem JavaScript. 1 ed. São Paulo: Novatec Editora, 2010

SILVA, João Augusto Marciano; OLIVEIRA, Cintia Carvalho. Análise avaliativa acerca do uso do React para programação web front-end. In: ENCONTRO DE ENSINO, PESQUISA E EXTENSÃO, 10., 2023, Patrocínio, MG. **Anais [...]**. Patrocínio, MG: IFTM, 2023.

SOUZA, Francisco Moreira Calado; LIMA, Edilson Carlos Silva; CARIDADE, Elda Regina de Sena. CRIANDO SISTEMA ESCALÁVEL DE AGENDAMENTOS UTILIZANDO TYPESCRIPT COM NESTJS NO BACKEND E NEXTJS NO FRONTEND. **Ibero-Americana de Humanidades, Ciências e Educação**, [S. l.], v. 8, n. 12, p. 43–57, 2022. DOI: 10.51891/rease.v8i12.7986.

SILVA, R. R.; RIVERO, L.; SANTOS, R. P. dos. ProgramSE: Um Jogo para Aprendizagem de Conceitos de Lógica de Programação. **Revista Brasileira de Informática na Educação**, [S. l.], v. 29, p. 301–330, 2021. DOI: 10.5753/rbie.2021.29.0.301.

SINCLAIR, Bruce. **IoT**: Como usar a "internet das coisas" para alavancar seus negócios. São Paulo: Autêntica Business, 2018. 240 p. Disponível em: https://www.google.com.br/books/edition/IoT_Como_Usar_a_Internet_Das_Coisas_Para/skIVDwAAQBAJ?hl=pt-BR&gbpv=0. Acesso em: 01 maio 2024.

VASCÓNCELOS, Joyciane Coelho; MOTTA LIMA, Patrícia Verônica Pinheiro Sales; ROCHA, Leonardo Andrade; KHAN, Ahmad Saeed. **Infraestrutura escolar e investimentos públicos em Educação no Brasil**: a importância para o desempenho educacional. Ensaio: Avaliação e Políticas Públicas em Educação, Rio de Janeiro, v. 29, n. 113, p. 874-898, out./dez. 2021.

VASCONCELLOS, M. S. de; CARVALHO, F. G. de; BARRETO, J. O.; ATELLA, G. C. As Várias Faces dos Jogos Digitais na Educação. **Informática na Educação**: teoria & prática, Porto Alegre, v. 20, n. 4, p. 203-218, ago. 2017.

VIANNA, M. et al. **Design Thinking**: Inovação em Negócios. 1. ed. Rio de Janeiro, RJ: MJV Press, 2012.

VILELA, João Paulo; LOPES, Ricardo; LIMA, Fernando. **Modelagem 3d de edifícios históricos**: a influência do lod no processo de reconstrução virtual. 2020. Trabalho de Graduação (Graduação em Engenharia Civil) - Universidade Estadual Paulista, Guaratinguetá, 2015.