2024
Escola Técnica 3D Colégios
RGI41
Professor: Gabriel Lyra
Aluna Luísa Matias de Pontes

Trabalho de Informática

Jogo da Forca - Documentação

O objetivo deste projeto é fazer o Jogo da forca utilizando HTML, CSS e PHP!!! HTML para a estrutura da página, PHP para a parte lógica e o CSS para o design. O jogo permite que o jogador tente adivinhar uma palavra, com o máximo de até 5 tentativas e tendo a chance de pedir 1 dica.

Instruções de uso.

1. Ao acessar o site, o jogo começa automaticamente com uma palavra aleatória.
2. O jogador desse adivinhar a palavra, utilizando letras.
3. O número de tentativas (erros) é limitado a 5.
4. O jogador pode pedir apenas 1 dica no jogo, podendo ver ela duas vezes.

Planejamento e Design.

Antes de desenvolver o jogo, eu decidi que ele deveria ter uma mecânica simples, com a principal funcionalidade sendo descobrir a palavra secreta. A lógica de erros seria controlada por variáveis de sessão.

No design do site, eu decidi que deveria ser algo simples, direto e agradável visualmente. Então as cores e os elementos que escolhi foram para proporcionar uma experiência de usuário intuitiva e confortável. Selecionei as minhas cores favoritas (vermelho, verde e roxo) e escolhi tons mais suaves para deixar o mais harmônico possível. Também adicionei um gradiente da cor creme para o branco no fundo para deixar bonito.

Escolhi a tipografia ‘Lato’ e ‘Poppins’ por elas não deixarem a leitura cansativa, e também para manter a atmosfera de acolhimento que eu queria transmitir para o meu site.
Além disso também escolhi bordas redondas para os botões e a barra de texto, mas eu explicarei mais a fundo isso na explicação do CSS.

Processo de desenvolvimento.

Estrutura do Código:

O código foi dividido entre os três componentes principais: HTML, CSS e PHP.

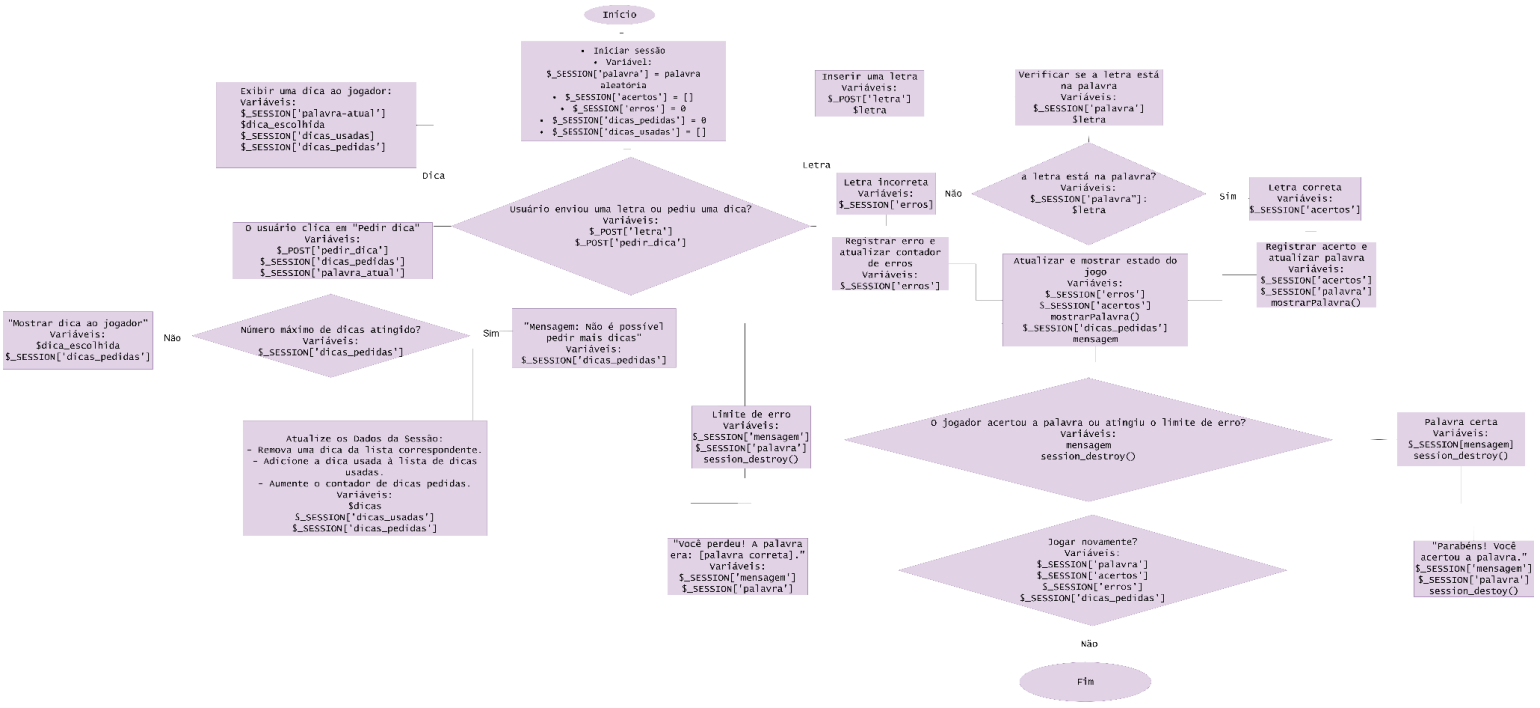
- HTML: Fornece a estrutura da página, incluindo a exibição da palavra a ser adivinhada, os campos de entrada e os botões.
- CSS: Define o estilo visual do jogo, incluindo cores, fontes e animações de interação, como a animação no botão feita nesse site.
- PHP: Responsável pela lógica do jogo, como verificar se a letra está correta, contar os erros, e controlar a quantidade de dicas usadas.

Lógica de Jogo:

Controle de sessão(PHP): O estado do jogo (palavra, acertos, erros e dicas) é mantido usando sessões ($_SESSION), permitindo que o jogador continue a partida após cada ação.
Função ‘mostrarPalavra()’: Essa função exibe a palavra com as letras acertadas e sublinhadas para as letras não descobertas.
Processo de Pedir Dica: O jogador pode clicar em um botão para pedir uma dica. O número de dicas é limitado a 1, podendo acessar a mesma dica 2 vezes, após passar do limite 2, o botão de Pedir Dica some.

Fluxograma:

Como a resolução do meu fluxograma não ficou tão boa, além da imagem, segue abaixo o link do meu fluxograma no draw.io
https://drive.google.com/file/d/18dIh7s6h-4HLz9vN3CbX_w4_HaiIUrSC/view?usp=sharing



O fluxograma acima representa as etapas de funcionamento do jogo, com as decisões que o usuário pode tomar, como enviar uma letra ou pedir uma dica. As variáveis associadas a cada etapa foram detalhadas para mostrar como o jogo processa as ações do usuário.

Portugol:

algoritmo "Jogo da Forca"

```
inicio
   // Declaração das variáveis
   palavras <- ["elefante", "computador", "programa", "felicidade", "amigo", "aviao", "musica",
"sol", "flor", "nuvem"]
   dicas <- { "elefante" -> ["Maior animal terrestre."],
            "computador" -> ["Máquina essencial para tecnologia moderna."],
            "programa" -> ["Pode ser criado com linguagens como Python ou Java."],
            "felicidade" -> ["Muitos a associam com conquistas ou momentos especiais."],
            "amigo" -> ["Pessoa com quem você compartilha confiança e afeto."],
            "aviao" -> ["Meio de transporte aéreo rápido."],
            "musica" -> ["Arte de combinar sons de forma harmoniosa."],
            "sol" -> ["Estrela central do sistema solar."],
            "flor" -> ["Muitas pessoas adoram receber como presente."],
            "nuvem" -> ["Pode anunciar chuva ou simplesmente embelezar o céu."] }

   // Variáveis do jogo
   palavra_atual <- palavras[aleatorio(0, tamanho(palavras)-1)] // Escolhe uma palavra
aleatória
   palavra_adivinhada <- lista_vazia
   erros <- 0
   acertos <- lista_vazia
   dicas_pedidas <- 0
   dicas_usadas <- lista_vazia
   dica_escolhida <- ""

   // Loop do jogo
   enquanto erros < 5 e tamanho(palavra_adivinhada) < tamanho(palavra_atual) faca
      escreva("Palavra: ", mostrar_palavra(palavra_atual, acertos))
      escreva("\nErros: ", erros, " de 5")
      escreva("\nDigite uma letra ou 'dica' para pedir uma dica: ")
      leia(entrada)

      se entrada == "dica" e dicas_pedidas < 2 entao
         dica <- pedir_dica(palavra_atual, dicas, dicas_usadas)
         escreva("Dica: ", dica)
         dicas_pedidas <- dicas_pedidas + 1
      senao
         se letra_na_palavra(entrada, palavra_atual) entao
            acertos <- adicionar_letra(acertos, entrada)
         senao
            erros <- erros + 1
         fim_se
      fim_se

      // Atualiza o estado do jogo
      escreva("\nPalavra: ", mostrar_palavra(palavra_atual, acertos))

      // Verificar se o jogador ganhou ou perdeu
```

```
        se palavra_completa(acertos, palavra_atual) entao
            escreva("\nParabéns! Você acertou a palavra: ", palavra_atual)
            pare
        senao se erros >= 5 entao
            escreva("\nVocê perdeu! A palavra era: ", palavra_atual)
            pare
        fim_se
    fim_enquanto

    // Perguntar se o jogador quer jogar novamente
    escreva("\nJogar novamente? (s/n): ")
    leia(jogar_novamente)
    se jogar_novamente == "s" entao
        // Reinicia o jogo
        reiniciar_jogo()
    senao
        escreva("Obrigado por jogar!")
    fim_se
fim_algoritmo

// Funções auxiliares

funcao mostrar_palavra(palavra_atual, acertos) retorne texto
    palavra_exibida <- ""
    para cada letra em palavra_atual faca
        se letra em acertos entao
            palavra_exibida <- palavra_exibida + letra + " "
        senao
            palavra_exibida <- palavra_exibida + "_ "
        fim_se
    fim_para
    retorne palavra_exibida
fim_funcao

funcao letra_na_palavra(letra, palavra_atual) retorne logico
    para cada letra_palavra em palavra_atual faca
        se letra == letra_palavra entao
            retorne verdadeiro
        fim_se
    fim_para
    retorne falso
fim_funcao

funcao adicionar_letra(acertos, letra) retorne lista
    acertos <- acertos + letra
    retorne acertos
fim_funcao
```

```
funcao palavra_completa(acertos, palavra_atual) retorne logico
   para cada letra em palavra_atual faca
      se letra nao esta em acertos entao
         retorne falso
      fim_se
   fim_para
   retorne verdadeiro
fim_funcao

funcao pedir_dica(palavra_atual, dicas, dicas_usadas) retorne texto
   dica <- dicas[palavra_atual][0] // Pega a primeira dica disponível
   dicas[palavra_atual] <- lista_remover(dicas[palavra_atual], 0) // Remove a dica utilizada
   dicas_usadas <- dicas_usadas + dica
   retorne dica
fim_funcao

funcao reiniciar_jogo()
   // Reinicia as variáveis e o jogo
   acertos <- lista_vazia
   erros <- 0
   dicas_pedidas <- 0
   dicas_usadas <- lista_vazia
   palavra_atual <- palavras[aleatorio(0, tamanho(palavras)-1)]
fim_funcao
```

Inicialização das variáveis:

- palavras: Lista de palavras que o jogo escolherá aleatoriamente.
- dicas: Dicas associadas a cada palavra.
- palavra_atual: Palavra que o jogador precisa adivinhar.
- acertos: Lista das letras que o jogador acertou.
- erros: Contador de erros do jogador.
- dicas_pedidas: Conta o número de dicas que o jogador pediu.
- dicas_usadas: Armazena as dicas que já foram usadas.
- dica_escolhida: Variável que guarda a dica que será apresentada ao jogador.

1. **Loop do jogo:**
2. O loop continua enquanto o número de erros for menor que 5 e a palavra não for completamente adivinhada.
3. O jogador insere uma letra ou escolhe pedir uma dica.

- **Verificação de letra:**
- Se o jogador acertar a letra, ela é adicionada aos acertos.
- Se errar, o número de erros é incrementado.

- **Verificação de vitória ou derrota:**
- O jogo verifica se a palavra foi completamente adivinhada ou se o jogador atingiu o limite de erros (5).

- Dicas:
- O jogador pode pedir até 2 dicas (no caso 1, mas pode ver 2 vezes). Quando uma dica é pedida, ela é removida da lista de dicas disponíveis e adicionada às dicas usadas.

- Reinício do jogo:
- O jogador pode escolher reiniciar o jogo após vencer ou perder.

Comentários do Código:

1. PHP:

---

session_start():

A função session_start() é iniciar uma nova sessão ou retomar uma sessão existente. Em um site, a sessão permite armazenar informações temporárias que podem ser acessadas em diferentes páginas ou momentos, sem a necessidade de pedir ao usuário para inserir informações repetidas vezes

---

```
$palavras = ['elefante', 'computador', 'programa', 'felicidade', 'amigo', 'aviao', 'musica', 'sol', 'flor', 'nuvem'];
```

Aqui, eu criei uma array chamada $palavras. Essa contém uma lista de palavras que o jogo da forca pode escolher para o jogador adivinhar.

```
$dicas = [
    'elefante' => ['Maior animal terrestre.'],
    'computador' => ['Máquina essencial para tecnologia moderna.'],
    'programa' => ['Pode ser criado com linguagens como Python ou Java.'],
    'felicidade' => ['Muitos a associam com conquistas ou momentos especiais.'],
    'amigo' => ['Pessoa com quem você compartilha confiança e afeto.'],
    'aviao' => ['Meio de transporte aéreo rápido.'],
    'musica' => ['Arte de combinar sons de forma harmoniosa.'],
    'sol' => ['Estrela central do sistema solar.'],
    'flor' => ['Muitas pessoas adoram receber como presente.'],
    'nuvem' => ['Pode anunciar chuva ou simplesmente embelezar o céu.']
```

```
];
```

A variável $dicas é um array associativo. Cada chave é uma palavra (como ‘computador’, ‘felicidade’, ‘sol’, etc.), e o valor é um array com dicas sobre essa palavra. As dicas ajudam o jogador a adivinhar a palavra no jogo da forca.

```
if (!isset($_SESSION['palavra'])) {
    $_SESSION['palavra'] = $palavras[array_rand($palavras)];
    $_SESSION['acertos'] = [];
    $_SESSION['erros'] = 0;
}
```

Esse bloco verifica se a variável de sessão $_SESSION[‘palavra’] está definida, caso não esteja, isso significa que o jogo está começando.

- Uma palavra aleatória é escolhida do array $palavras com a função array_rand().
- Inicializa o array $_SESSION[‘acertos’] (para armazenar as letras que o jogador acertou) e variável $_SESSION[‘erros’] (que conta os erros do jogador).

```
if (!isset($_SESSION['palavra_atual'])) {
    $_SESSION['palavra_atual'] = $_SESSION['palavra'];
}
```

Aqui, a variável de sessão $_SESSION[‘palavra_atual’] é definida com a palavra que foi selecionada anteriormente. Isso ajuda a manter a palavra atual do jogo disponível para uso em outros lugares do código.

```
if (!isset($_SESSION['dicas_pedidas'])) {
    $_SESSION['dicas_pedidas'] = 0;
}

if (!isset($_SESSION['dicas_usadas'])) {
    $_SESSION['dicas_usadas'] = [];
}
```

Esse bloco inicializa as variáveis que controlam as dicas:

- $_SESSION[‘dicas_pedidas’] guarda o número de dicas que o jogador já pediu. Inicialmente, esse valor é 0.
- $_SESSION[‘dicas_usadas’] é um array que armazena as dicas que já foram usadas.

```
if (isset($_POST['pedir_dica'])) {
    if ($_SESSION['dicas_pedidas'] < 2) {
        $palavra_atual = $_SESSION['palavra_atual'];
        if (isset($dicas[$palavra_atual]) && count($dicas[$palavra_atual]) > 0) {
```

```
            $dica_escolhida = array_pop($dicas[$palavra_atual]);
            $_SESSION['dicas_usadas'][] = $dica_escolhida;
            $_SESSION['dicas_pedidas']++;
        }
    }
}
```

Esse bloco verifica se o jogador pediu uma dica (através do botão “Pedir Dica”).

- Se o número de dicas pedidas for menor que 2 ($_SESSION[‘dicas_pedidas’] <2), o código irá verificar se ainda existem dicas para a palavra atual, escolher uma dica aleatória (usando array_pop()), adicionar essa dica ao array de dicas já usadas e incrementar o contador de dicas pedidas.

```
function mostrarPalavra() {
    $palavra = $_SESSION['palavra'];
    $acertos = $_SESSION['acertos'];
    $resultado = ' ';

    foreach (str_split($palavra) as $letra) {
        if (in_array($letra, $acertos)) {
            $resultado .= $letra . ' ';
        } else {
            $resultado .= '_ ';
        }
    }
    return rtrim($resultado);
}
```

Essa função exibe a palavra atual no estado em que ela se encontra (com letras acertadas ou underscores/sublinhadas para as letras que ainda não foram acertadas).

O loop percorre a cada letra da palavra e, se ela foi acertada, exibe a letra. Caso contrário, exibe um _ no lugar da letra.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['letra']) &&
!empty($_POST['letra'])) {
    $letra = strtolower($_POST['letra']);

    if (strpos($_SESSION['palavra'], $letra) !== false) {
        if (!in_array($letra, $_SESSION['acertos'])) {
            $_SESSION['acertos'][] = $letra;
        }
    } else {
        $_SESSION['erros']++;
    }

    $palavraCompleta = str_replace(' ', '', mostrarPalavra());
    if ($palavraCompleta === $_SESSION['palavra']) {
```

```
        $mensagem = "Parabéns! Você acertou a palavra: {$_SESSION['palavra']}";
        session_destroy();
    }

    if ($_SESSION['erros'] >= 5) {
        $mensagem = "Você perdeu! A palavra era: {$_SESSION['palavra']}";
        session_destroy();
    }
}
```

Esse bloco de código verifica se o jogador enviou uma letra.
Se a letra estiver na palavra, ela é adicionada aos acertos. Caso contrário, o contador de erros é incrementado.

- A função mostrarPalavra() é chamada para verificar se o jogador completou a palavra, comparando as letras acertadas com a palavra completa. Se o jogador acertou, o jogo termina com uma mensagem de vitória.
- Se o número de erros atingir 5, o jogador perde, e o jogo também termina.

Exibição do Resultado:
O código exibe mensagens como “Parabéns! Você acertou a palavra!” ou “Você perdeu! A palavra era…” dependendo do resultado final.

---

2. HTML:

Estrutura básica e fundamental:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jogo da Forca!</title>
    <link
href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&family=Poppins:wght@400;700&display=swap" rel="stylesheet">
```

`<!DOCTYPE html>`: Declara que o documento está no padrão HTML5.
`<html lang="pt-BR">`: Define o idioma como português do Brasil.
`<meta charset="UTF-8">`: Define a codificação como UTF-8 para suportar caracteres como "ç" e acentos.
`<meta name="viewport"`: Torna o site responsivo, ajustando-se a diferentes tamanhos de tela.

<link>: Importa uma fonte do Google Fonts, personalizando o visual do texto.

```
<body>
   <h1>Jogo da Forca</h1>
   <div class="letras">
      <?php echo mostrarPalavra(); ?>
   </div>
   <p>Erros: <?php echo $_SESSION['erros']; ?>/5</p>
```

<h1>: Exibe o título principal do jogo. E o título foi centralizado com uma fonte suave (Poppins) que dá destaque ao nome do jogo sem exagero.
<div class=”letras: Mostra a palavra com as letras descoberta ou “_”.
<p>: Mostra o número de erros cometidos e o limite máximo.

```
<form action="" method="POST">
   <label for="letra">Digite uma letra:</label><br>
   <input type="text" name="letra" maxlength="1" required><br>
   <button type="submit">Tentar</button><br>
</form>
```

- **<form>:** Envia dados ao servidor.
  - **action="":** Envia os dados para a mesma página.
  - **method="POST":** Usa o método POST para segurança e transferência de dados.
- **<label>:** Descreve o campo de entrada de texto.
- **<input>:** Campo para o jogador inserir uma letra. Define:
  - **name="letra":** A chave usada para acessar os dados no PHP.
  - **maxlength="1":** Restringe a entrada a apenas um caractere.
  - **required:** Torna o preenchimento obrigatório.
- **<button>:** Botão para enviar o formulário.

Fiz com que o campo de entrada (Input) tivesse bordas arredondas e também deixei q o botão utilizasse cores claras harmoniosas para reforçar a ideia de um visual mais aconchegante.

A organização em colunas mantém a clareza do layout.

```
<p>Erros: <?php echo $_SESSION['erros']; ?>/5</p>

<?php if (isset($_POST['pedir_dica']) && !empty($dica_escolhida)) { ?>

   <p class="mensagem">Dica: <?php echo $dica_escolhida; ?></p>

<?php } ?>
```

As mensagens foram posicionadas logo abaixo da palavra, para garantir que o jogador veja imediatamente o estado atual do jogo.

A mensagem de erro usa cores contrastantes, como o vermelho claro, para transmitir atenção sem ser agressiva.

As dicas são exibidas em um tom suave, reforçando o apoio ao jogador.

```
<button type="submit" name="pedir_dica" id="botaodica">Pedir Dica</button>
```

Os botões com bordas arredondas e animação de “crescer” ao passar o mouse dão uma experiência mais divertida.

A cor verde pastel no botão representa tranquilidade e encoraja a interação.

```
<?php if ($_SESSION['dicas_pedidas'] < 2) { ?>

<form action="" method="POST">

    <button type="submit" name="pedir_dica">Pedir Dica</button>

</form>

<?php } ?>
```

- **Condicional PHP:** Verifica se o jogador ainda pode pedir dicas antes de exibir o botão.
- **`name="pedir_dica"`:** Identifica que este formulário solicita uma dica.

```
<?php if (isset($mensagem)) { ?>

<p class="mensagem"><?php echo $mensagem; ?></p>

<a href="index.php" class="replay-link">Jogar novamente</a>

<?php } ?>
```

Exibe uma mensagem de vitória ou reinicia o jogo.

CSS:

```
body {
    background: linear-gradient(to bottom, #F4F1DE, #FFFDF7);
    color: #333333;
}
```

- O fundo gradiente claro, indo da cor creme para o branco, criando uma sensação de acolhimento e simplicidade
- O texto usa cinza escuro, que é neutro e fácil de ler.

```
font-family: 'Lato', sans-serif;
font-family: 'Poppins', sans-serif;
```

- A combinação das fontes **Lato** e **Poppins** foi escolhida para reforçar a modernidade e aconchego.
- Poppins é usada para títulos maiores, enquanto o Lato aparece nos textos de instrução.

```
button {
    background-color: #81B29A;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 10px;
    transition: background-color 0.3s, transform 0.2s;
}
button:hover {
    background-color: #6A994E;
    transform: scale(1.05);
}
```

- O botão em verde claro combina com o tema suave, incentivando a interatividade.
- A animação de transformação (aumentando agilmente o tamanho no hover) mantém o jogador engajado.
- Cores contrastantes no botão (fundo verde claro e texto branco) aumentam a legibilidade e atratividade.

```
input[type="text"] {
    border: 2px solid #A8D5BA;
    border-radius: 20px;
    font-size: 1rem;
    transition: all 0.3s ease-in-out;
}
input[type="text"]:focus {
    border-color: #6A994E;
    box-shadow: 0 0 8px rgba(106, 153, 78, 0.6);
}
```

- As bordas arredondadas do campo de entrada reforçam o visual amigável.
- Ao focar no campo, a animação de brilho verde indica prontidão para a interação, transmitindo calma.

```
.letras {
    font-size: 2rem;
    letter-spacing: 0.3rem;
    font-family: 'Courier New', monospace;
    color: #81B29A;
}
```

Usei uma fonte monoespaçada (Courier New) para simular o estilo tradicional do jogo da forca. A cor verde clara que usei combina com o tema e é visualmente relaxante.

Consideração Final.

Este projeto foi uma experiência interessante para mim, pude unir a lógica de programação e o design em um só trabalho. Desde o início, meu objetivo foi criar algo simples, funcional, e visualmente muito agradável, algo mais em um estilo aconchegante e acolhedor para o usuário. Por isso, eu escolhi cores suaves que combinam com a simplicidade do jogo.

Cada detalhe foi bem pensado, como as animações dos botões, as bordas arredondas nos elementos e até a escolha das fontes, que passam uma sensação de leveza e harmonia. Na parte lógica, eu me esforcei para manter o código organizado e eficiente, utilizando sessões para gerenciar o estado do jogo.

No final, fiquei muito satisfeita com o resultado. Este trabalho me fez ter a oportunidade de poder unir a minha criatividade com programação, além de me desafiar a buscar soluções que fossem funcionais e esteticamente agradáveis. Estou orgulhosa do que consegui criar e ansiosa para continuar aprendendo e desenvolvendo projetos ainda melhores.