Programação Web Coding
Tema: Buscador de receitas

Alunos: Welligton (04192751)
Thauan Levyr (04193411)
Brenno José (04196107)

Este site foi criado para facilitar a vida de quem ama cozinhar ou simplesmente precisa preparar algo rápido e prático no dia a dia. Ele funciona como um buscador completo de receitas, reunindo pratos variados para todos os gostos, ocasiões e níveis de experiência na cozinha.

Aqui, você encontra desde receitas simples para o cotidiano até opções mais elaboradas para surpreender em momentos especiais. Cada prato é apresentado com a lista completa de ingredientes, detalhes do modo de preparo e, para tornar tudo ainda mais fácil, vídeos explicativos que mostram o passo a passo de forma visual e intuitiva.

O objetivo é tornar a culinária acessível, prática e divertida, permitindo que qualquer pessoa possa cozinhar com confiança. Seja para aprender algo novo, testar sabores diferentes ou encontrar aquela receita perfeita de última hora, este buscador é o seu aliado ideal na cozinha.

Explicação do código

1. O HTML — A estrutura do site

O HTML é como o “esqueleto” da página. Ele organiza tudo que vai aparecer na tela.

Principais partes desse HTML:

Topo – Menu

```html
<header>
    <nav class="navbar">
        <a href="#" id="btn-home">Início</a>
        <a href="#" id="btn-sobre">Sobre</a>
        <a href="#" id="btn-contato">Contato</a>
    </nav>
</header>
```

- Isso cria o menu lá em cima.
- Os links são usados para trocar entre as páginas **Home**, **Sobre** e **Contato**.

Seções principais:

O site tem **três páginas**, mas todas dentro do mesmo arquivo (isso é comum em sites simples).

Seção Home:

```html
<section id="sec-home">
    <h1>Buscador de Receitas</h1>
    <form class="search-form">
        <input type="text" placeholder="Digite o Ingrediente">
        <button>Buscar</button>
    </form>
    <div class="recipe-list"></div>
    <div class="recipe-details"></div>
</section>
```

Aqui temos:

- Uma caixa para digitar o ingrediente
- Uma área para mostrar as receitas encontradas
- Uma área para mostrar os detalhes da receita escolhida

Seção Sobre:

```html
<section id="sec-sobre" style="display:none;">
```

- Essa seção começa escondida.
- Ela só aparece quando o usuário clica em "Sobre".

Seção Contato:

Mesma lógica da seção "Sobre". Fica escondida até clicarem no menu.

Rodapé:

```html
<footer>
    <p>© 2025 WELL RECEITAS – <a href="#" id="btn-sobre-footer">Saiba mais</a></p>
</footer>
```

Tem um link que também abre a página "Sobre".

2. O CSS — A aparência do site

O CSS é responsável **pelo visual**: cores, tamanhos, bordas, espaçamento etc.

Principais coisas que ele faz:

Define o visual do site:

```css
body {
    font-family: Arial;
    background-color: #f4f4f4;
}
```

Faz o menu ficar fixo no topo:

```css
.navbar {
    position: fixed;
    top: 0;
    width: 100%;
}
```

Isso deixa o menu sempre visível, mesmo quando rola a página.

Cria os cards das receitas:

```css
.recipe-card {
    width: 150px;
    border-radius: 8px;
    transition: 0.3s;
}
.recipe-card:hover {
    transform: scale(1.05);
}
```

Quando passa o mouse, aumenta um pouco

3. JavaScript — O cérebro do site

O JavaScript faz o site funcionar de verdade:

- Trocar de página
- Buscar receitas na API
- Criar os cards dinamicamente
- Mostrar detalhes da receita

A. Troca entre as páginas ("Home", "Sobre", "Contato")

Pegando os botões do menu:

```js
const btnHome = document.getElementById('btn-home');
const btnSobre = document.getElementById('btn-sobre');
const btnContato = document.getElementById('btn-contato');
```

Pegando as seções:

```js
const secHome = document.getElementById('sec-home');
const secSobre = document.getElementById('sec-sobre');
const secContato = document.getElementById('sec-contato');
```

Função que esconde tudo e mostra apenas uma seção:

```js
function showSection(section) {
    secHome.style.display = 'none';
    secSobre.style.display = 'none';
    secContato.style.display = 'none';
    section.style.display = 'block';
}
```

Quando clico no menu → troca de página:

```js
btnHome.addEventListener('click', () => showSection(secHome));
btnSobre.addEventListener('click', () => showSection(secSobre));
btnContato.addEventListener('click', () => showSection(secContato));
```

B. Sistema de busca de receitas

Quando o usuário digita um ingrediente e clica em “Buscar”, acontece isto:

1 Captura o envio do formulário:

```js
form.addEventListener('submit', async (e) => {
    e.preventDefault();
```

`e.preventDefault()` impede o site de recarregar.

2 Pega o ingrediente digitado:

```js
const ingredient = e.target[0].value.trim();
```

3 Faz a busca na API de receitas:

```js
const res = await fetch(`https://www.themealdb.com/api/json/v1/1/filter.php?i=${ingredient}`);
const data = await res.json();
```

API é um serviço que devolve informações — nesse caso, receitas.

4 Se encontrou receitas → mostra na tela:

```js
showRecipes(data.meals);
```

C. Criando os cards (as receitas na lista)

A função abaixo cria vários cardzinhos de receita:

```js
function showRecipes(recipes) {
    recipeList.innerHTML = recipes.map(meal => `
        <div class="recipe-card" data-id="${meal.idMeal}">
            <img src="${meal.strMealThumb}">
            <h3>${meal.strMeal}</h3>
        </div>
    `).join('');
```

- `recipes.map()` percorre todas as receitas
- Cria um card para cada uma
- E coloca na tela

D. Quando o usuário clica em um card:

```js
card.addEventListener('click', async () => {
    const id = card.dataset.id;
    const res = await fetch(`https://www.themealdb.com/api/json/v1/1/lookup.php?i=${id}`);
    const meal = (await res.json()).meals[0];
    showDetails(meal);
});
```

- A API traz a receita completa
- Depois chama `showDetails()` para mostrar tudo

E. Mostrando detalhes da receita:

```js
function showDetails(meal) {
    recipeDetails.innerHTML = `
        <h2>${meal.strMeal}</h2>
        <img src="${meal.strMealThumb}">
        <h3>Ingredientes:</h3>
        <ul>
            ${getIngredients(meal).map(i => `<li>${i}</li>`).join('')}
        </ul>
        <h3>Modo de Preparo:</h3>
        <p>${meal.strInstructions}</p>
    `;
}
```

Mostra:

- Foto
- Nome
- Ingredientes
- Modo de preparo
- Link pro vídeo (se existir)

F. Pegando os ingredientes

A API envia ingredientes separados assim:

- strIngredient1
- strIngredient2
- strIngredient3
  etc...

Por isso o código precisa montar a lista:

```js
function getIngredients(meal) {
    const ingredients = [];
    for (let i = 1; i <= 20; i++) {
        const ing = meal[`strIngredient${i}`];
        const measure = meal[`strMeasure${i}`];
        if (ing && ing.trim()) ingredients.push(`${ing} - ${measure}`);
    }
    return ingredients;
}
```

Ele verifica de 1 até 20 e guarda só os que não estão vazios.