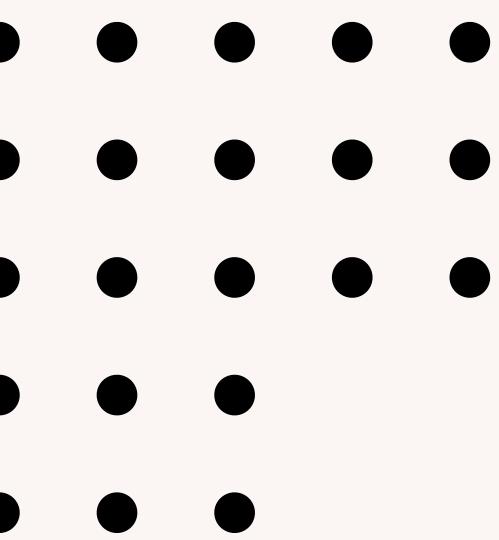
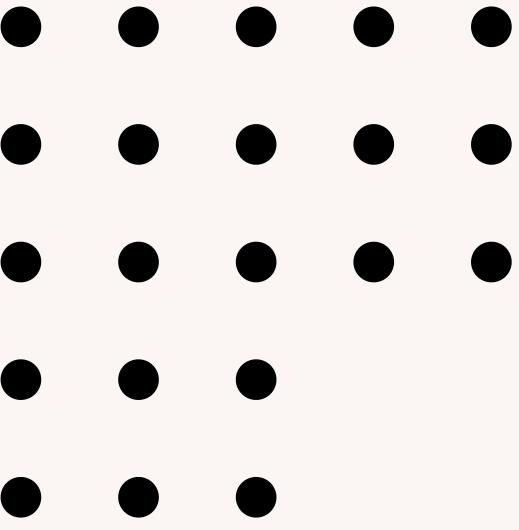
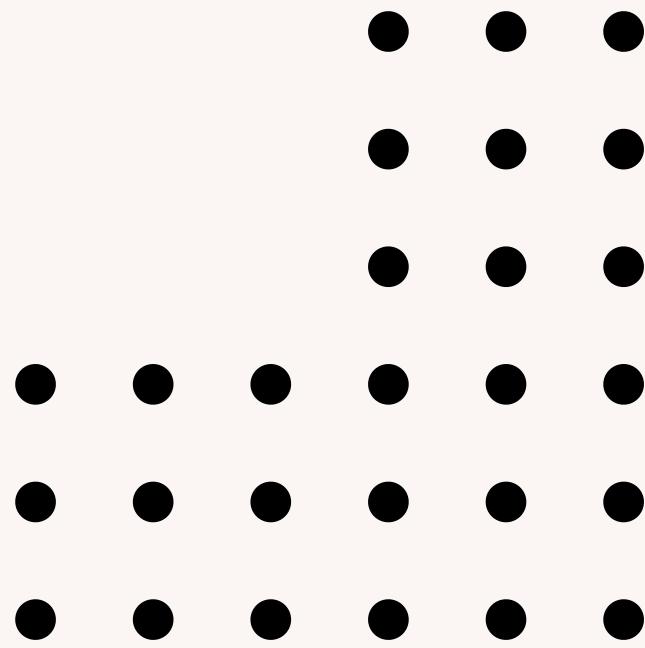


---

# **USABILIDADE, DEV WEB, MOBILE E JOGOS - AIMORES NOITE-AULA 7**



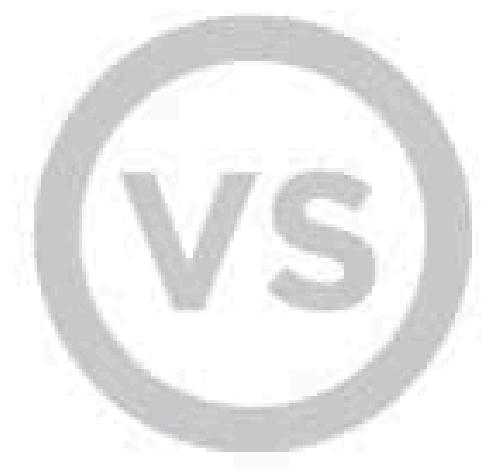
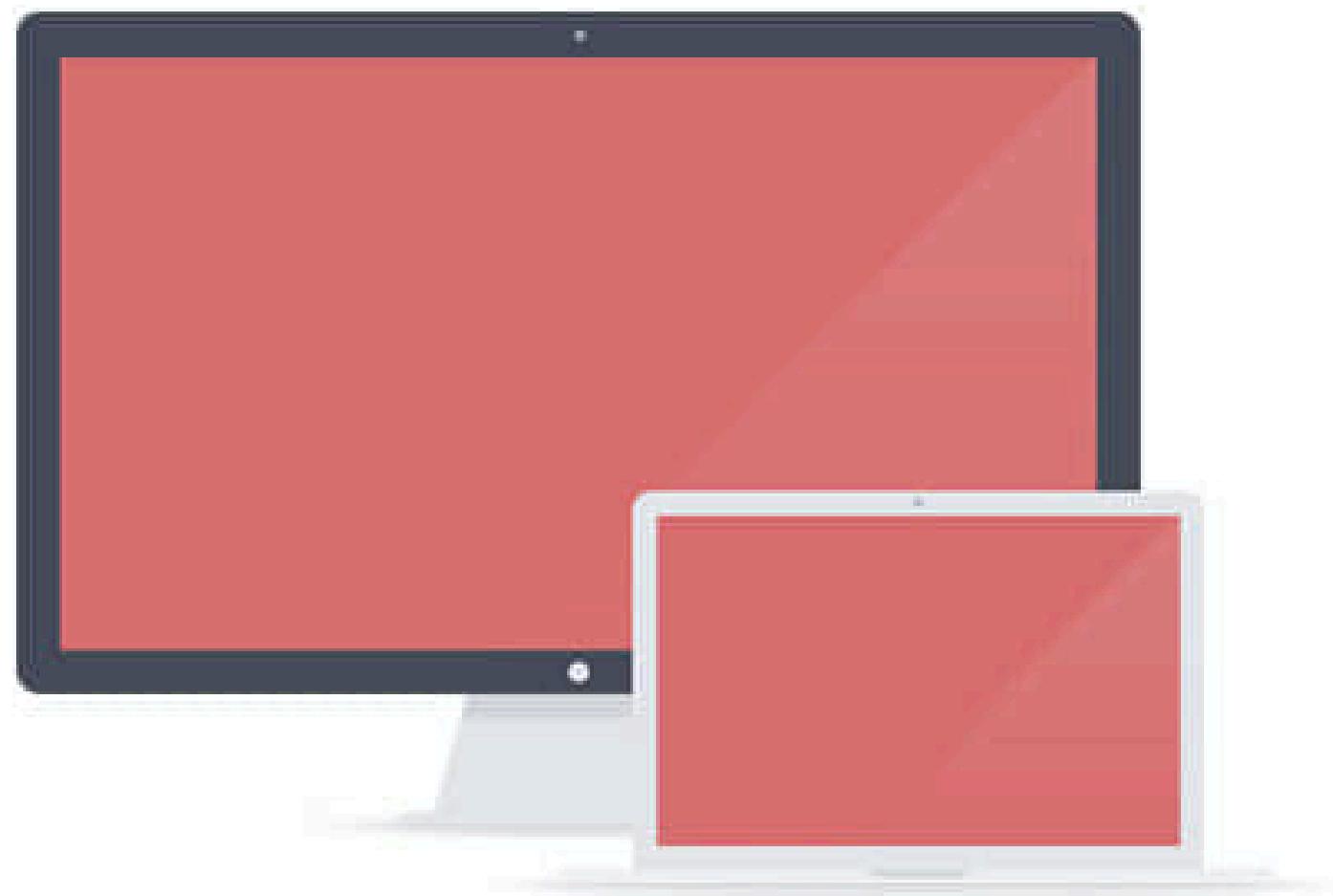
Prof: João Vitor do Vale



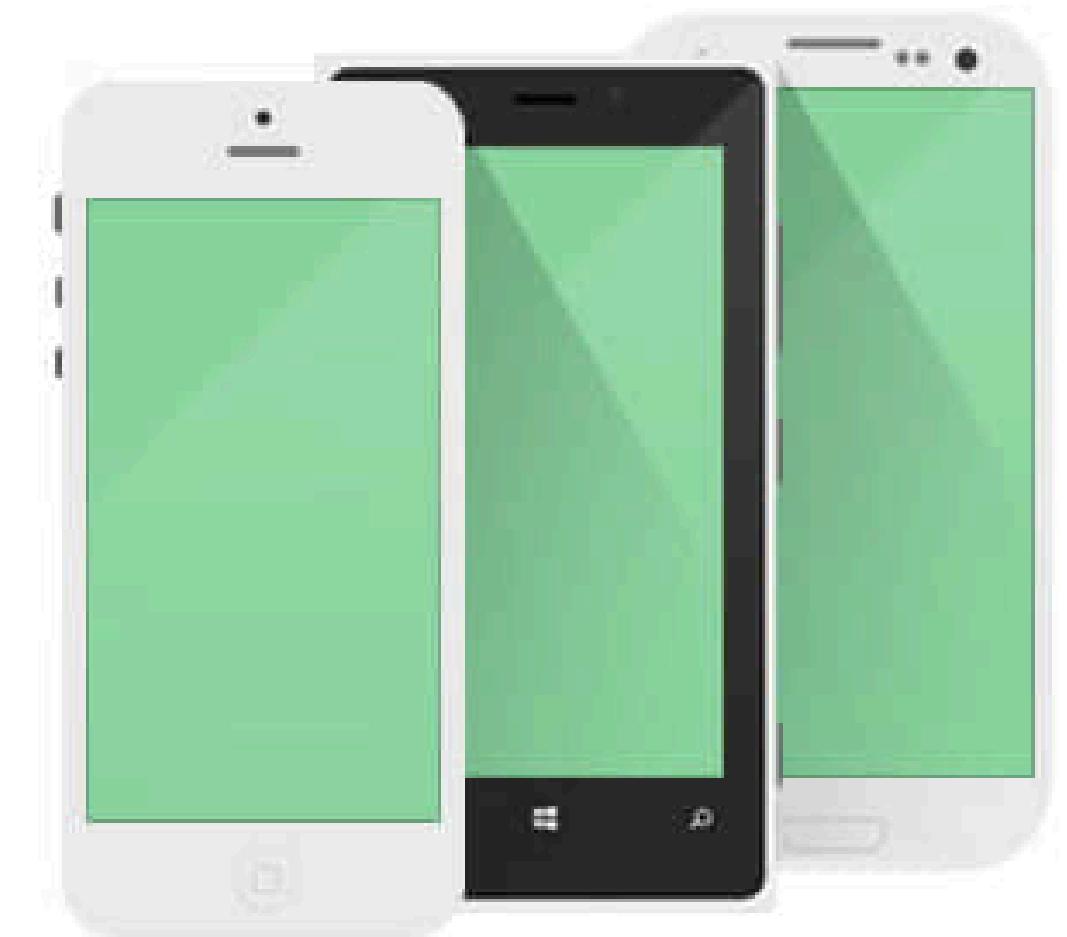
# **MOBILE FIRST**

**REDUZIR OU ADICIONAR  
CONTEÚDO?**

**DESKTOP**

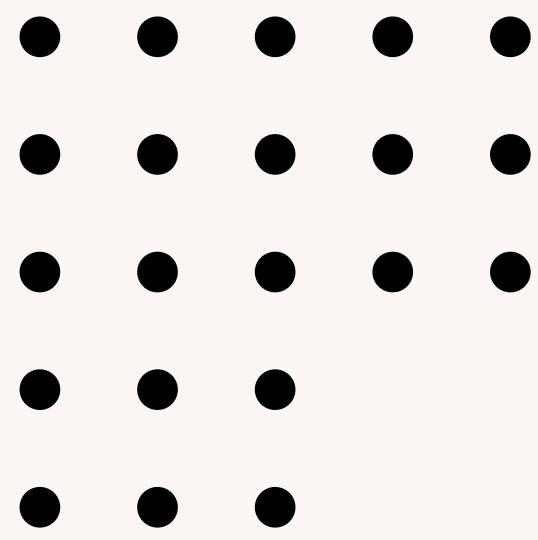


**MOBILE**



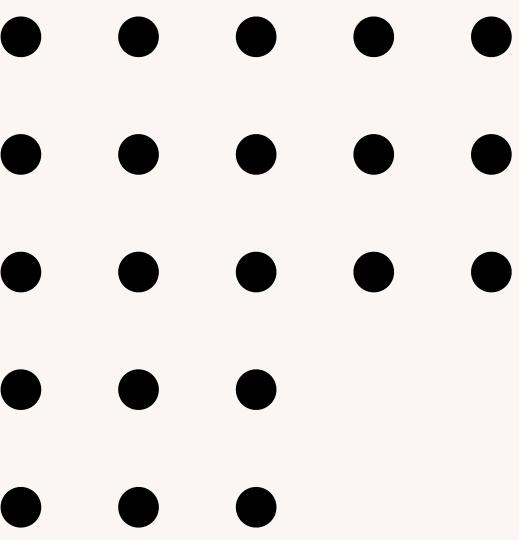
---

O conceito Mobile First surgiu em 2010 quando o diretor de produto do Google afirmou que o desenvolvimento de sites e plataformas digitais deveria ser feito pensando, primeiramente, nas necessidades mobile.



---

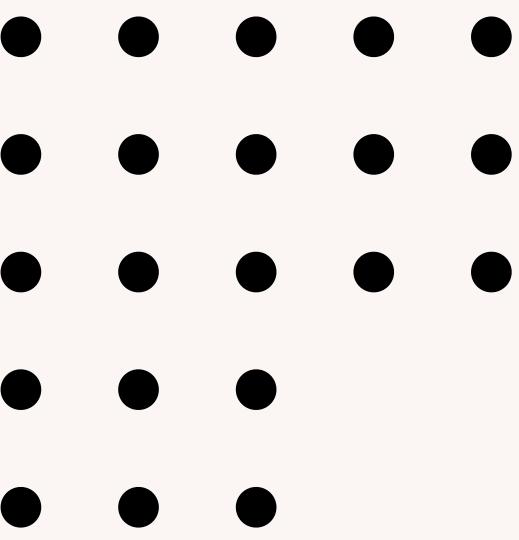
Desde então, o e-commerce passou a correr atrás de aplicativos e buscou um melhor desempenho dos seus sites em dispositivos móveis, uma vez que, segundo pesquisa do Instituto de inteligência Ipsos, 94% dos brasileiros fazem compras ou pagamentos utilizando apps mobile.



---

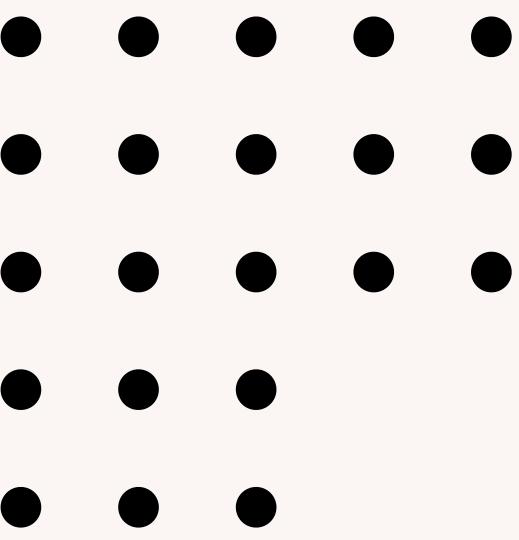
## Relação entre Mobile First e Mobile Responsive

Mobile Responsive ou, na tradução crua, “móvel responsivo”, é quando um site ou app é desenvolvido pensando apenas na adaptação dele aos dispositivos móveis. Ao contrário do Mobile First, o Mobile Responsive não propõe uma total otimização para tablets ou smartphones.



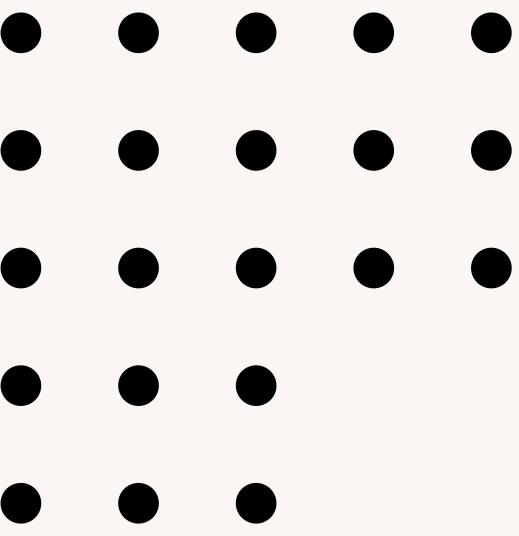
---

Ou seja, o conteúdo de um site que tem o design responsivo exibido em mobiles é exatamente o mesmo do desktop para melhorar a experiência do usuário. O conceito só flerta com uma formatação diferente.



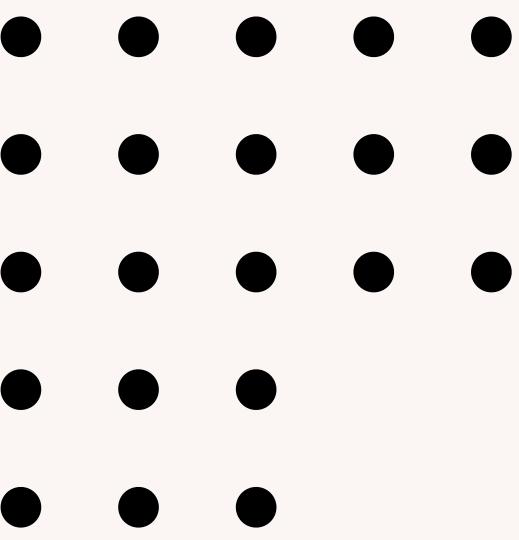
---

Segundo o Instituto de pesquisas Ipsos, 25% dos comerciantes brasileiros ainda não pensam no mercado mobile, não possuem sites responsivos e nem apps. Essa porcentagem demonstra como muitas empresas ainda precisam investir no design mobile.



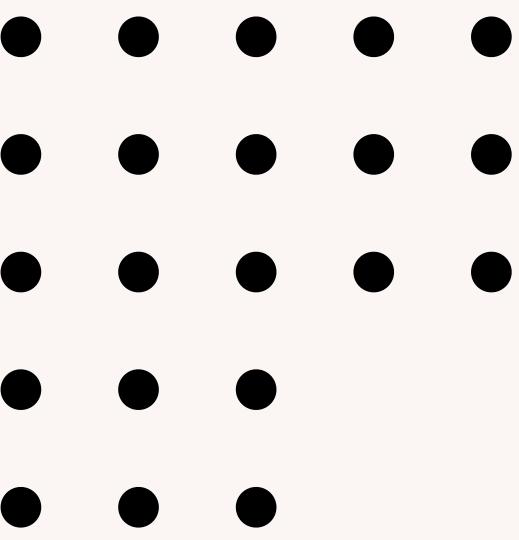
---

No final de 2016, o Google anunciou o lançamento gradual e o teste de um novo algoritmo móvel chamado Mobile First. Isso significa que o Google passou a colocar ainda mais foco nos dispositivos móveis como uma “entidade autônoma” e começou a oferecer benefícios de classificação para a criação de site responsivo.

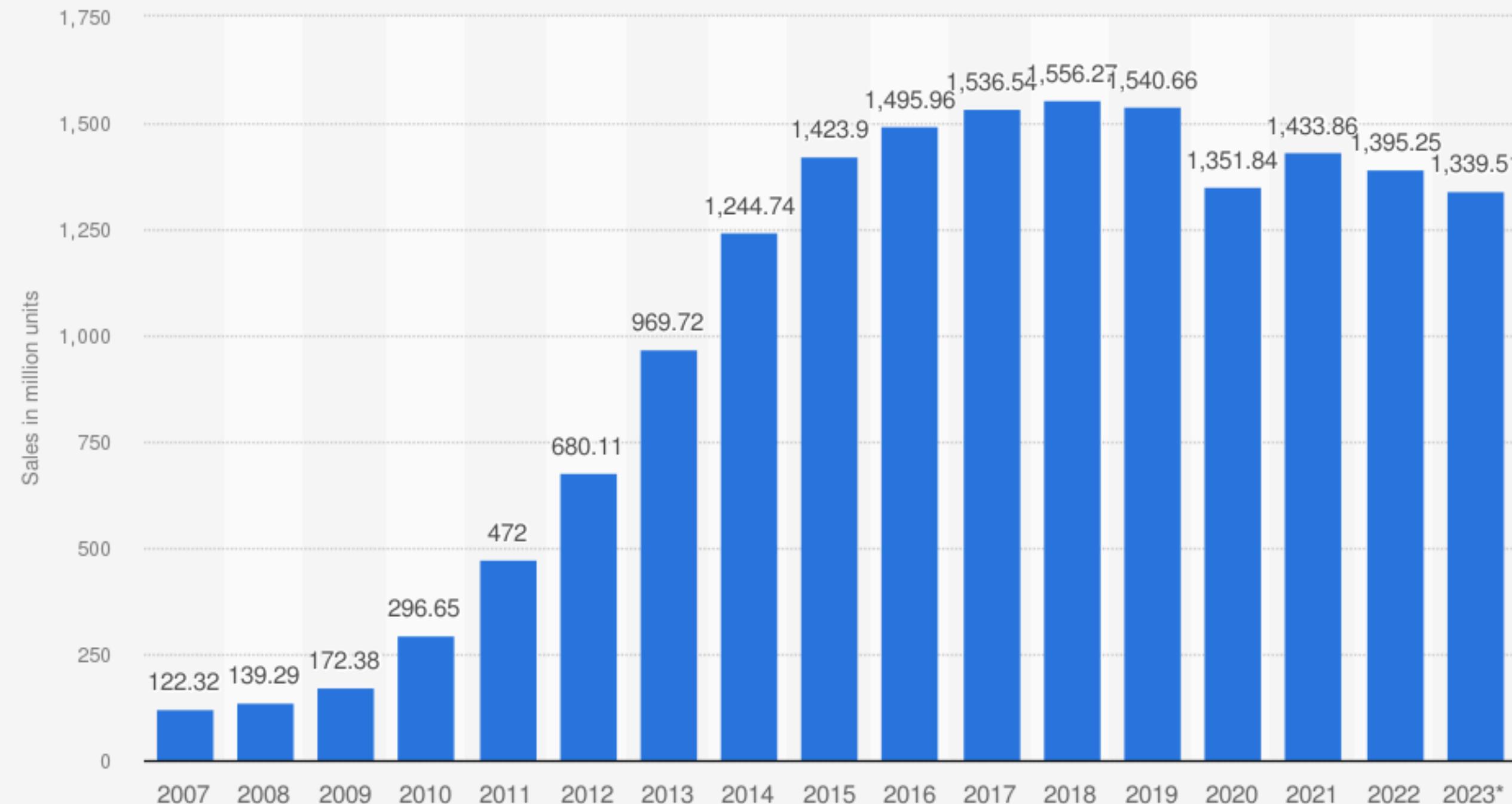


---

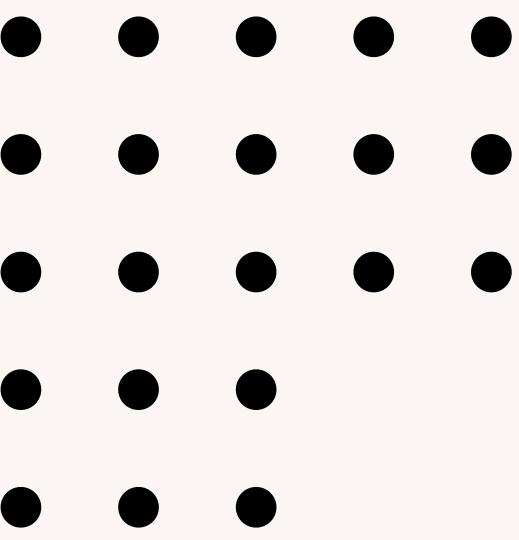
Sendo assim, como a tendência é que tudo esteja sempre mudando e novas coisas surgindo, é importante entender ao certo o que significa o Mobile First e quais as informações que o próprio Google forneceu a respeito desse algoritmo.



## Number of smartphones sold to end users worldwide from 2007 to 2022 (in million units)



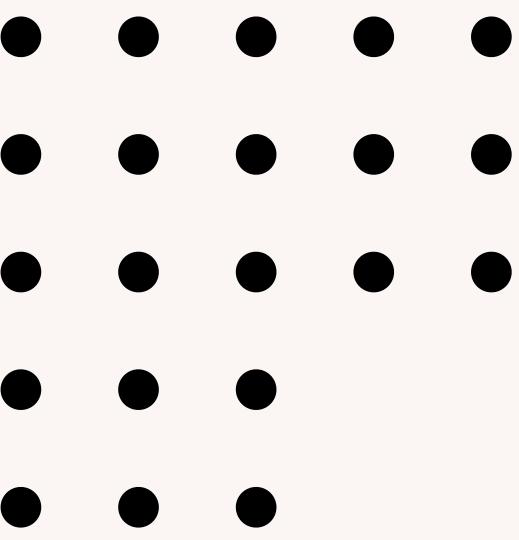
“Nossos sistemas de classificação geralmente ainda analisam a versão desktop do conteúdo de uma página para avaliar sua relevância para o usuário. Isso pode causar problemas quando a página para dispositivos móveis tem menos conteúdo, porque nossos algoritmos não estão avaliando a página real que é vista por um pesquisador móvel.” - **Primeiro parágrafo do blog Mobile First do Google**



---

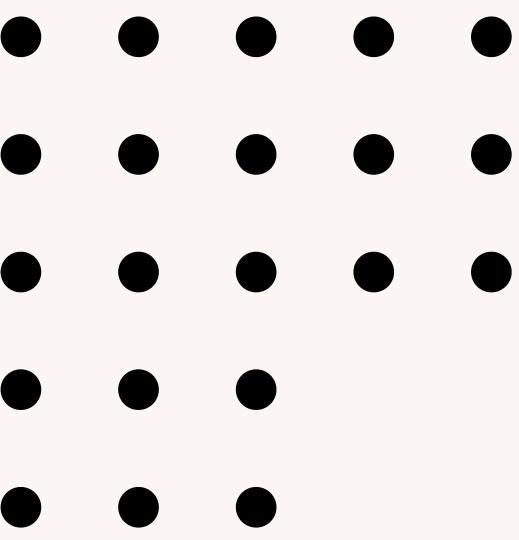
## Qual é o processo feito hoje em dia para os projetos web?

Atualmente, a maioria das empresas desenvolve os seus sites pensando no desktop como tela principal. A partir disso, o conteúdo é adaptado para atender a demanda dos dispositivos móveis. Com o mobile first, os papéis se invertem.



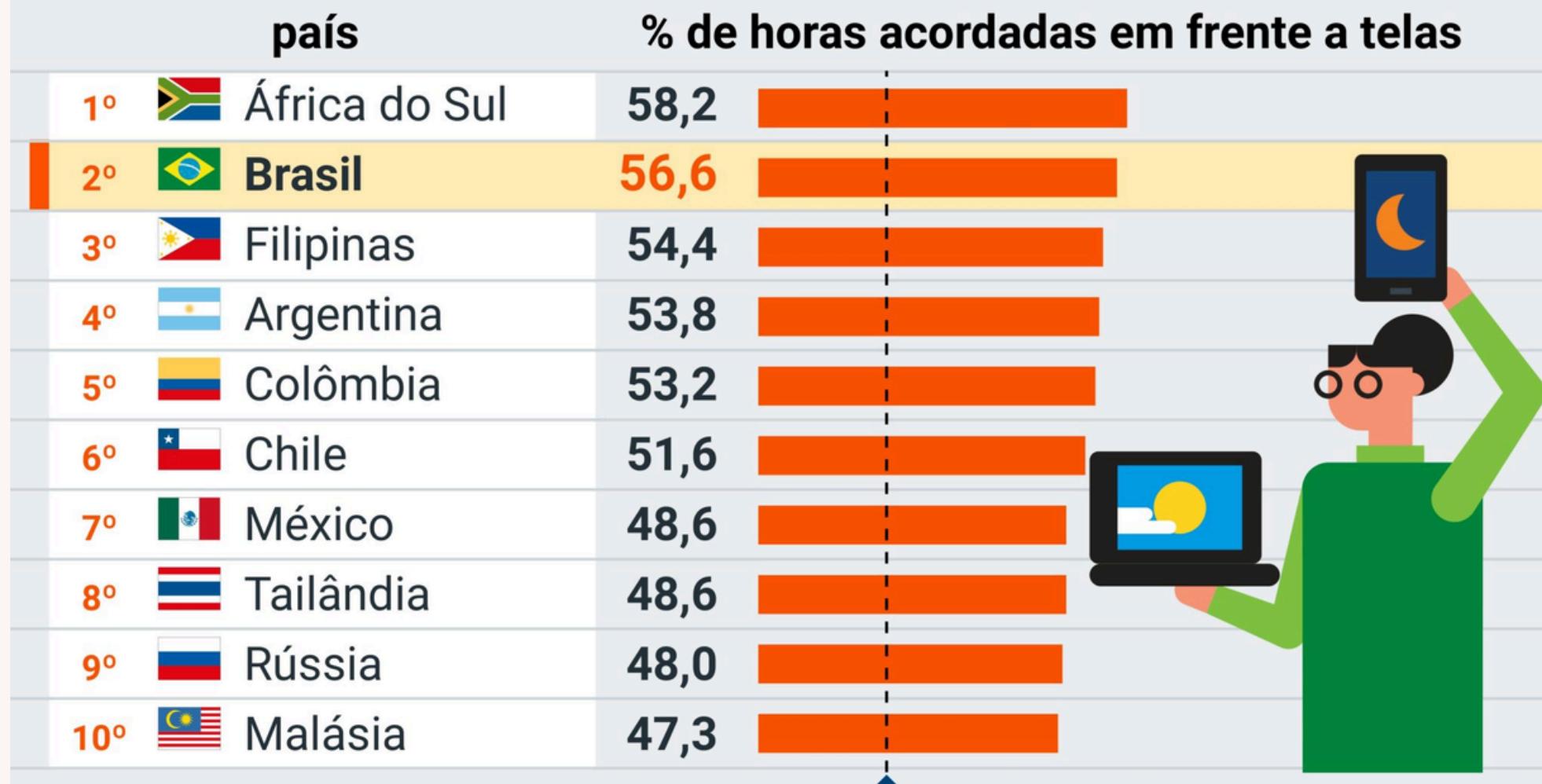
---

O smartphone se transformou na primeira tela – que já é a bastante tempo, desde 2014 – e os computadores recebem uma derivação do que foi arquitetado para o mobile. A partir de agora, quem pensa em investir em estratégias digitais deve realizar qualquer planejamento a partir das menores telas, favorecendo, assim, a experiência do usuário em termos de performance e usabilidade da página, além do melhor acesso ao conteúdo.



# BRASIL É O 2º PAÍS COM MAIOR TEMPO DE TELA DO MUNDO

% de horas acordadas gastas por pessoas em frente a telas



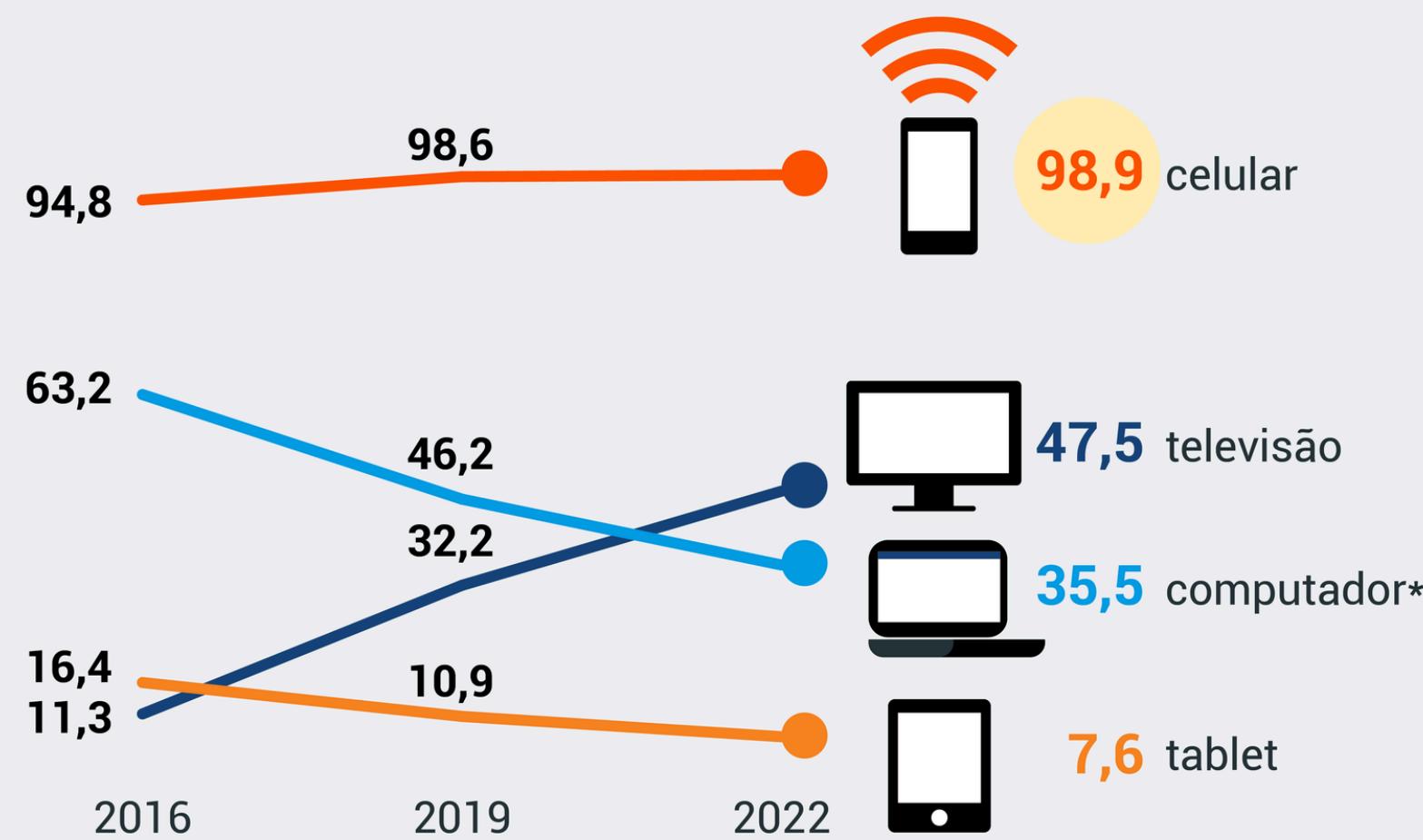
fonte: Electronics Hub

**PODER**  
360

Além disso, o Brasil também é o 2º país que mais gasta tempo nas redes sociais. Das 9 horas em que os brasileiros estão conectados, 4 são destinadas a navegar em plataformas como Instagram e Facebook.

## ACESSO À INTERNET NO BRASIL POR PESSOAS DE 10 ANOS OU MAIS

por equipamento utilizado (% da população)



obs.: percentuais se referem a 161,6 milhões de pessoas com 10 anos ou mais de idade que utilizaram a internet no Brasil em 2022

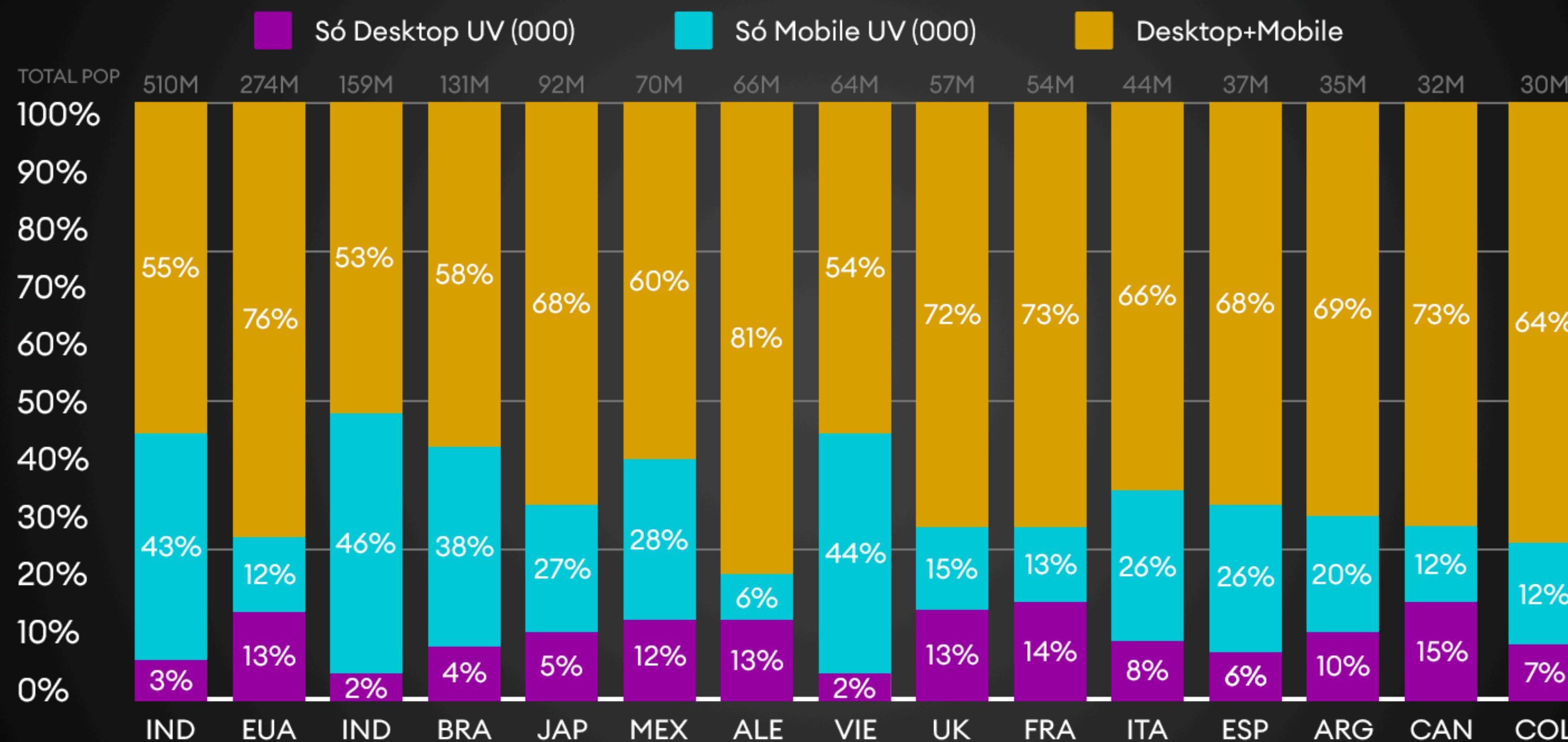
\*laptop ou de mesa

fonte: Pnad Contínua TIC Pessoas – 2022, do IBGE

O número de domicílios no Brasil com acesso à internet chegou a 68,9 milhões em 2022, o equivalente a 91,5%. É o que mostra a pesquisa TIC (Tecnologia da Informação e Comunicação) da Pnad (Pesquisa Nacional por Amostra de Domicílios Contínua), divulgada pelo IBGE (Instituto Brasileiro de Geografia e Estatística).

# Brasil segue como um país mobile first.

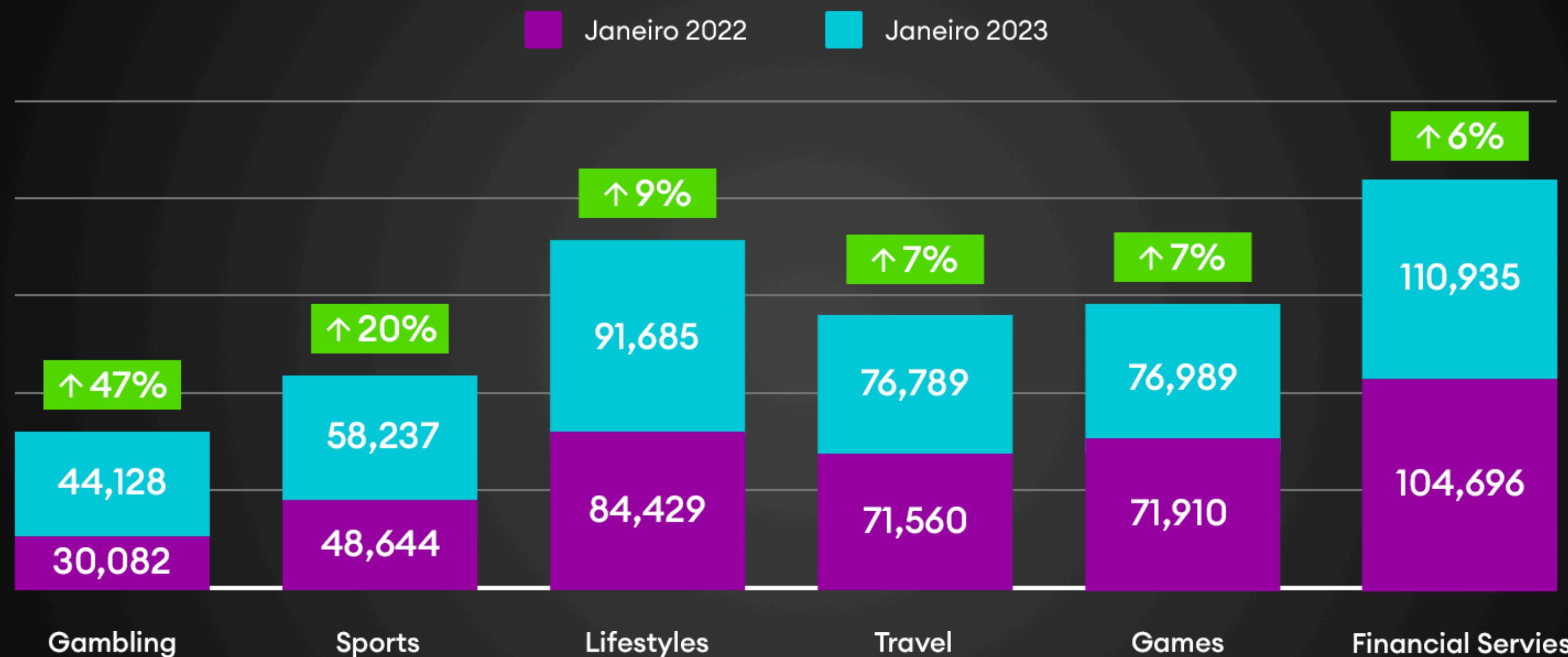
US e Alemanha são os países que mais consomem conteúdo em múltiplas telas



Comscore MMX Multi-Plataform - Multi-Market Key Measures  
Total Internet - Dez/22 - Múltiplos países

F

# Top categorias que mais cresceram exclusivamente em mobile



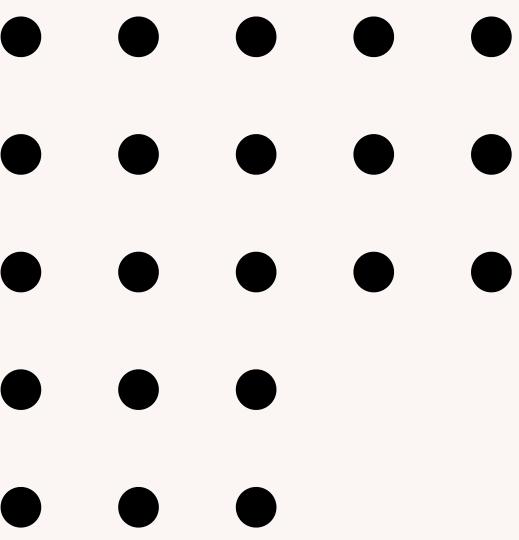
Fonte: Comscore MMX Multi-Plataform - Key Measures  
Uvs e Total Minutos - Jan/22 e Jan/23 - Brasil

F

---

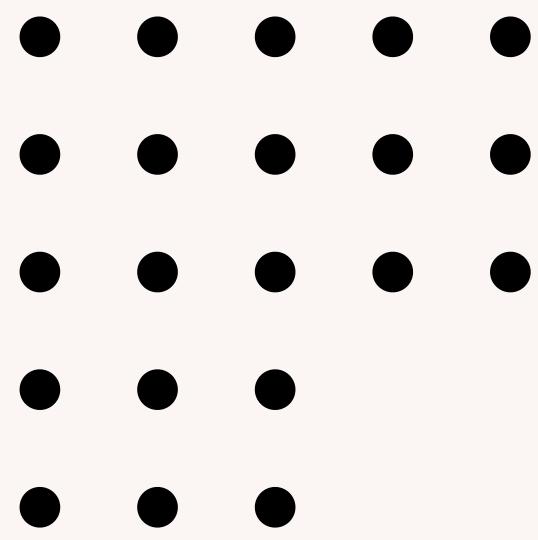
## A importância de se desenvolver uma estratégia Mobile First

Antes de mais nada, vamos a um dado específico sobre o usuário de internet no Brasil. Os smartphones são usados por 99% das pessoas conectadas no país, sendo que 59% acessam a internet apenas pelo celular. Se considerarmos o público das classes D e E, o acesso exclusivo pelo smartphone chega a 85%.



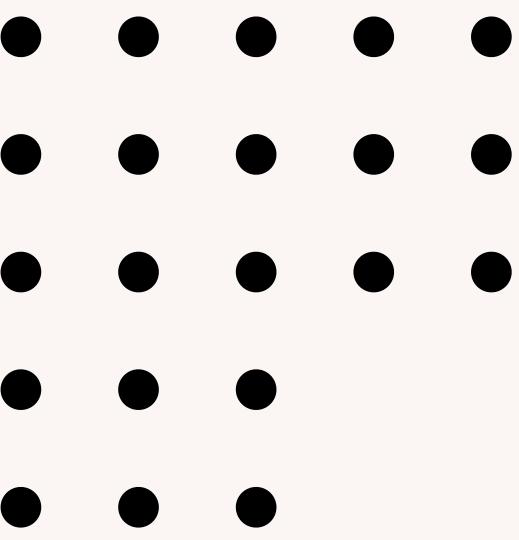
---

Dessa forma, as empresas precisam entender, de uma vez por todas, que seu público tem um smartphone à mão e, cada vez mais, prioriza o uso de tal dispositivo para se conectar.



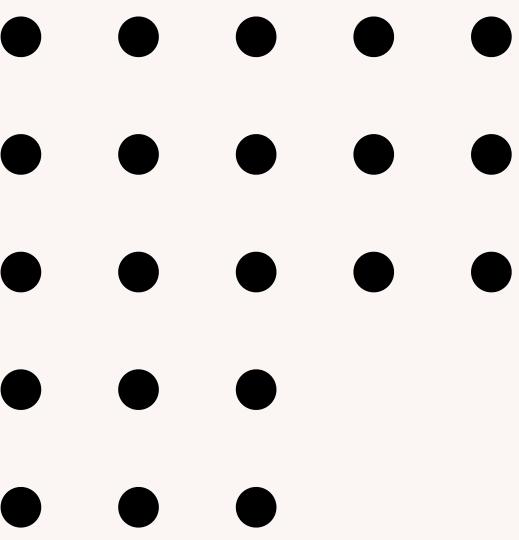
## Quais são as vantagens do mobile first?

- Melhor ranqueamento no Google
- Melhor experiência do usuário
- Novos canais de receita
- Melhor relacionamento e retenção do cliente
- Aumento da credibilidade da marca
- Otimização do carregamento das páginas



---

**HTML + CSS**  
**Atv Rápida**



## CSS

Estiliza o site

Adiciona estilo para os elementos da página

Segmenta vários tipos de tela para que a página fique responsiva

Lida primeiramente com a visualização da página

## HTML

Cria a estrutura do site

Controla o layout do conteúdo

Fornece uma estrutura para os padrões da [web](#)

É o fundamento básico para criar qualquer tipo de site

## Javascript

Aumenta a interatividade

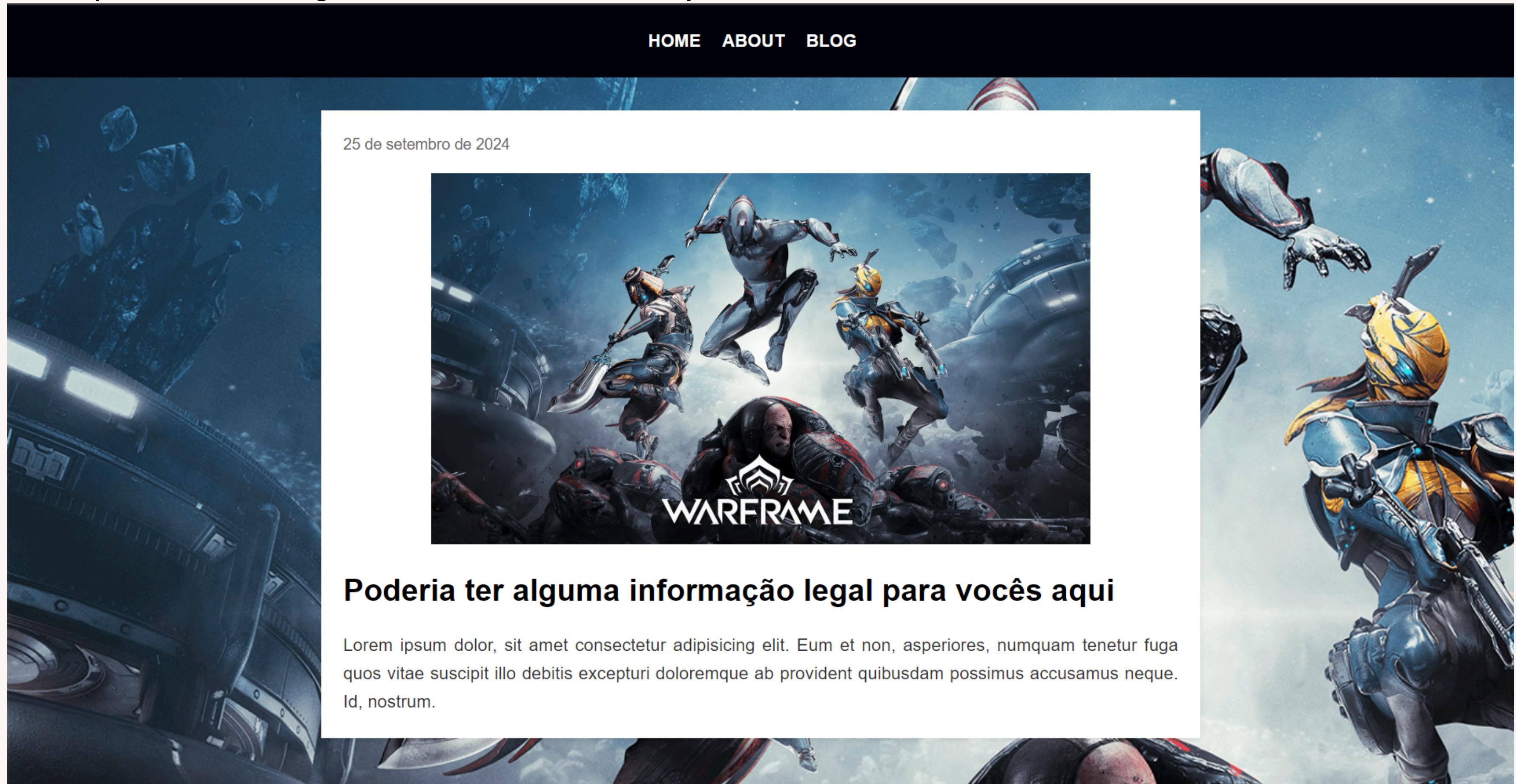
Adiciona interatividade no site

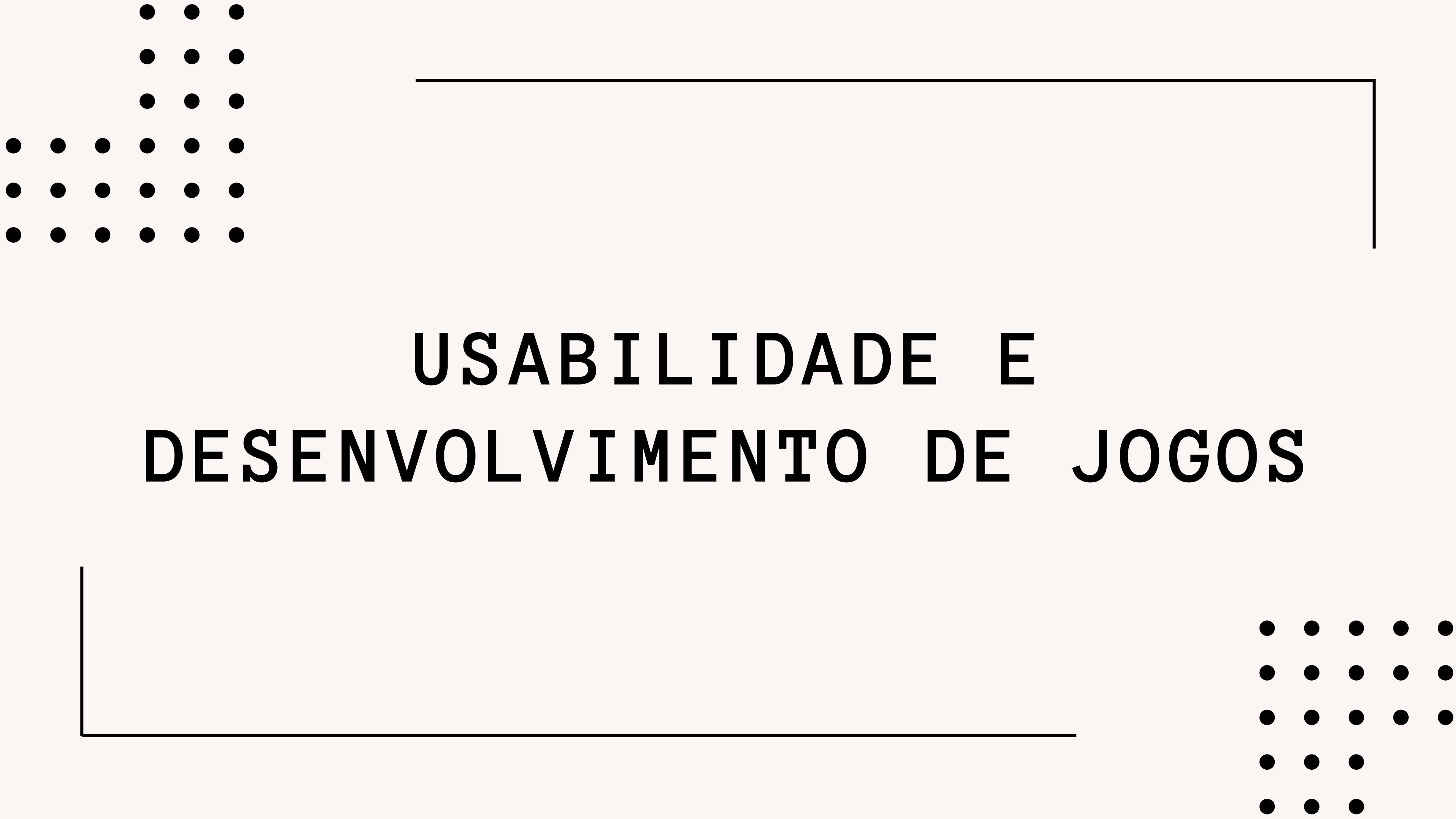
Lida com complexas funções e lógicas

Linguagem de programação que melhora a funcionalidade da aplicação

Prática em sala 1.

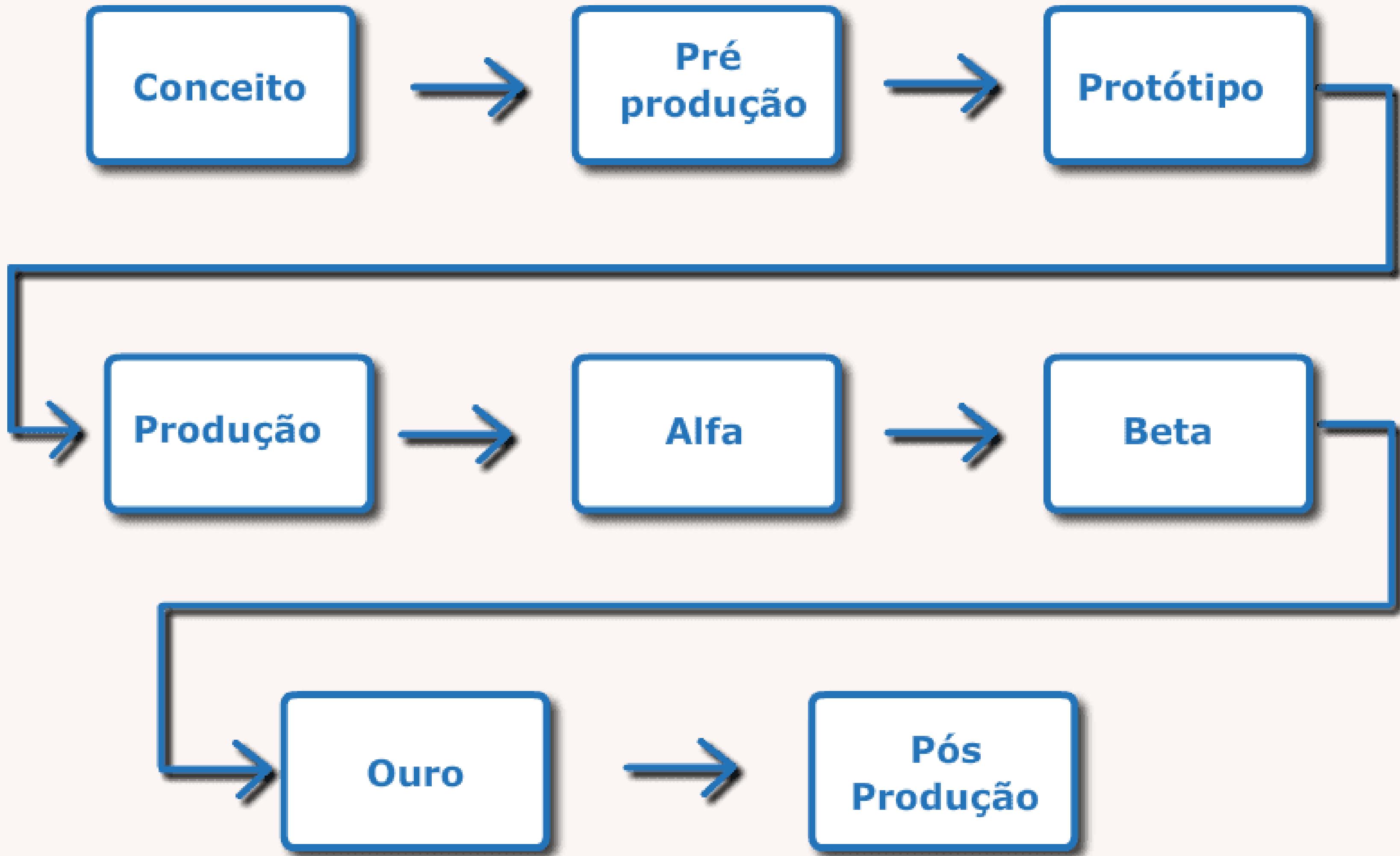
Reproduza o seguinte site utilizando apenas html e css:





# **USABILIDADE E DESENVOLVIMENTO DE JOGOS**





# PROCESSO DE CRIAÇÃO DE JOGOS

## **Conceito:**

- Esta fase inicial ocorre na elaboração da ideia de game, sua funcionalidade básica, o enredo, a viabilidade econômica etc. Esta etapa termina quando se toma a decisão de começar a planejar o projeto de jogo.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Conceito:

O ideal, neste ponto, é gerar um documento de conceito de jogo com, no máximo, cinco páginas, que deve conter os seguintes componentes:

- Premissa
- Diferenciais do jogo
- Perfil do público-alvo
- Tipo ou gênero do jogo
- Classificação etária
- Plataforma-alvo e requisitos de hardware
- Licenças e autorizações de terceiros
- Análise de competitividade
- Objetivos do game
- Perspectiva de retorno financeiro.

# PROCESSO DE CRIAÇÃO DE JOGOS

## **Pré-produção ou planejamento:**

- Fase com detalhamento do jogo, estilo de arte, plano de produção, level design, mecânicas de jogo etc. Nesta etapa participam programadores, artistas, escritores, designers, produtores e assistentes de produtores.

# PROCESSO DE CRIAÇÃO DE JOGOS

## **Pré-produção ou planejamento:**

- Segundo Novak (2011), deve-se generalizar uma estrutura considerando os itens:
- Interface do game
- Habilidade e itens dos personagens
- Mundo do game e motor (engine) pretendido para a produção.

# PROCESSO DE CRIAÇÃO DE JOGOS

## **Protótipo:**

- O próximo objetivo das equipes de desenvolvimento de um game é criar um protótipo concreto. O protótipo pode ser produzido em formato digital, em papel ou qualquer outros recurso que apresente a ideia e mecânicas pretendidas para o jogo.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Protótipo:

- Entretanto, antes de ingressar no reino virtual, certamente é útil criar um protótipo de baixa fidelidade (geralmente em papel, usando cartões, painéis, ladrilhos e/ou miniaturas) do game e testá-lo internamente para garantir que a mecânica do modo de jogar é impecável e o game é divertido e atraente.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Produção:

- Depois que o protótipo for aprovado, a equipe de desenvolvimento deverá estar pronta para entrar na fase mais extensa: a produção, quando o game é efetivamente desenvolvido. Essa fase geralmente dura de 6 meses a 2 anos, resultando em um game acabado.

# PROCESSO DE CRIAÇÃO DE JOGOS

## **Alfa:**

- A fase alfa é o ponto em que um game pode ser jogado do começo ao fim. Talvez haja algumas lacunas e os elementos artísticos não sejam definitivos, mas o motor e a interface do usuário estão completos. Em vez da construção e criação, a fase alfa é voltada para o acabamento e os ajustes finais no game

# PROCESSO DE CRIAÇÃO DE JOGOS

## **Alfa:**

- Durante a fase alfa, o departamento de testes testa cada módulo do game pelo menos uma vez, cria um banco de dados de defeitos e um plano de testes e registra os defeitos e os resultados dos testes de desempenho. Testadores temporários de jogabilidade são incorporados à equipe nessa fase para localizar problemas.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Alfa:

Para passar pela fase alfa, os seguintes elementos do game deverão estar concluídos:

- Percurso completo de jogo (ou seja, ele é jogável do começo ao fim);
- Texto no idioma básico;
- Interface básica,
- Documentação preliminar;
- Requisitos mínimos do sistema testados;
- Maioria das interfaces manuais testadas quanto à compatibilidade;
- Arte e áudio temporários;
- Recursos multijogador testados (quando aplicável);
- Rascunho do manual do game.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Beta:

- Depois de passar pela fase alfa, o game entra na fase beta. Nessa fase, a ênfase está na correção de problemas. Todos os materiais já foram incorporados ao game, e o processo de produção terminou. Na fase beta, o objetivo é estabilizar o projeto e eliminar o maior número possível de defeitos antes que o produto comece a ser vendido

# PROCESSO DE CRIAÇÃO DE JOGOS

## Beta:

- Os objetivos dessa fase incluem a detecção de todos os defeitos e problemas de desempenho significativos; a realização de testes completos, com correção dos problemas e ajustes de desempenho; e a realização de testes em todas as plataformas compatíveis.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Beta:

- Os seguintes elementos deverão estar concluídos para que o game passe pela fase beta:
  - código;
  - conteúdo;
  - texto em diferentes idiomas;
  - navegação pelo percurso do game;
  - interface do usuário;
  - compatibilidade de hardware e de software;
  - compatibilidade da interface manual;
  - arte e áudio;
  - manual do game.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Ouro:

- Tendo passado pela fase beta, o game entra na fase ouro. Ele é enviado para o fabricante depois que um dos discos mestres é testadometiculamente e considerado aceitável. Na fase ouro, a administração superior já avaliou o produto e o banco de dados de defeitos (bugbase) e concordou que ele está pronto.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Ouro:

- A fabricação dura várias semanas, durante as quais a mídia é gerada e embalada. Ao sair da fase ouro, o game é lançado no mercado. É cada vez mais comum que a fase de fabricação seja totalmente eliminada, porque a popularidade dos downloads digitais vem reforçando a tendência das vendas on-line (virtuais) em lugar do fornecimento de produtos físicos (reais).

# PROCESSO DE CRIAÇÃO DE JOGOS

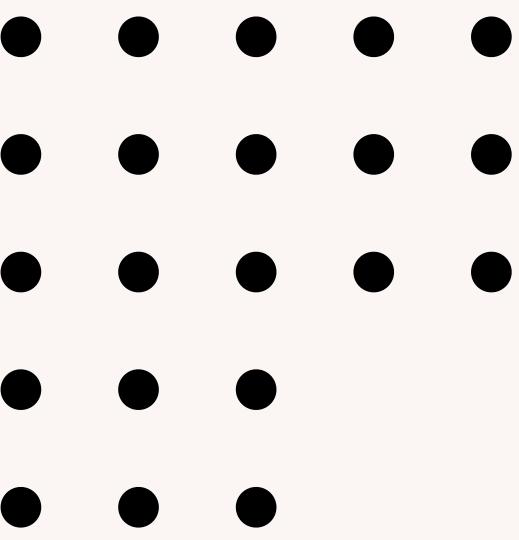
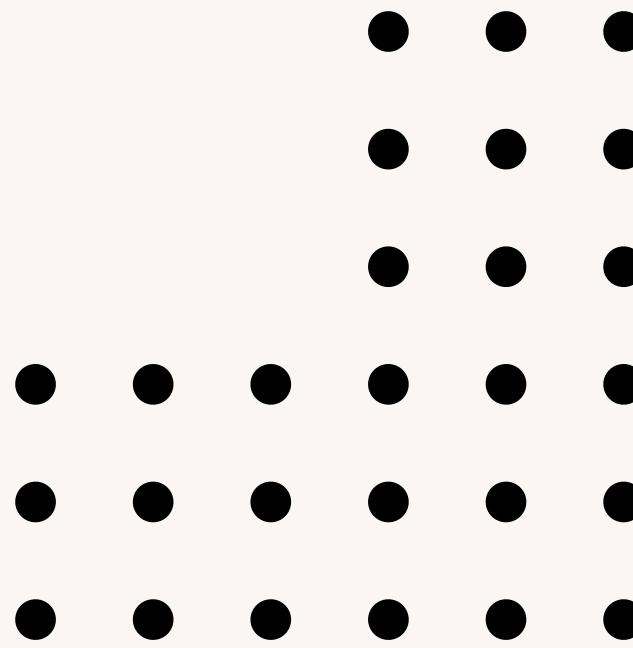
## Pós-produção:

- Durante a fase de pós-produção (ou pós-lançamento), várias versões subsequentes também podem ser lançadas para substituir e melhorar o game original, aumentando sua longevidade. Essas novas versões são oferecidas gratuitamente e criadas por meio de aplicação de correções, atualizando-o com conteúdos adicionais que aprimoram o game original.

# PROCESSO DE CRIAÇÃO DE JOGOS

## Pós-produção:

- Uma correção também pode ser aplicada para corrigir falhas de programação — que não são necessariamente problemas graves e, às vezes, solucionam pequenas dificuldades, como fazer o game funcionar adequadamente em uma configuração de hardware incomum. As atualizações — conteúdos adicionais criados para aprimorar o game original — também são lançadas durante essa fase



# **ATIVIDADE PRÁTICA – JUMP GAME JS**

**H T M L**

# — ATIVIDADE PRÁTICA – JUMP GAME JS

## Jump Game:

- Dentre as possíveis abordagens para a criação de um jogo simples em JS, aqui se escolheu por utilizar o arquivo **JS** fora do arquivo **HTML** principal, da mesma forma que ocorre com o **CSS**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Jump Game</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  
  <div class="game">
    <div id="character"></div>
    <div id="block"></div>
  </div>
  <p>Pontuação: <span id="scoreSpan">0</span></p>
  <script src="script.js"></script>
</body>
</html>
```

# — ATIVIDADE PRÁTICA – JUMP GAME JS

Jump Game:

Passo 1: Estruturando a página com HTML

O HTML serve como a base do jogo, fornecendo a estrutura e os elementos que usaremos.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Jump Game</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    
    <div class="game">
      <div id="character"></div>
      <div id="block"></div>
    </div>
    <p>Pontuação: <span id="scoreSpan">0</span></p>
    <script src="script.js"></script>
  </body>
</html>
```

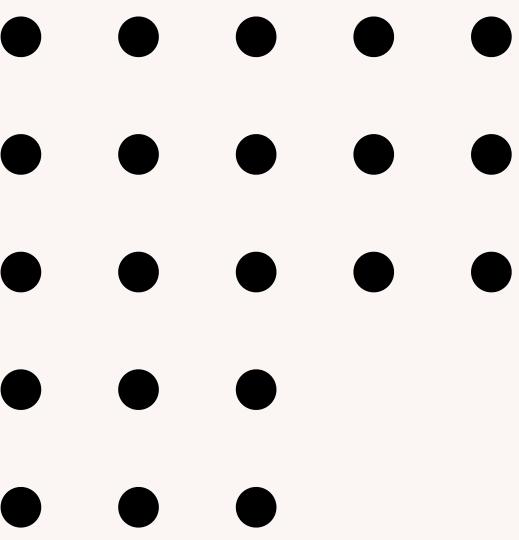
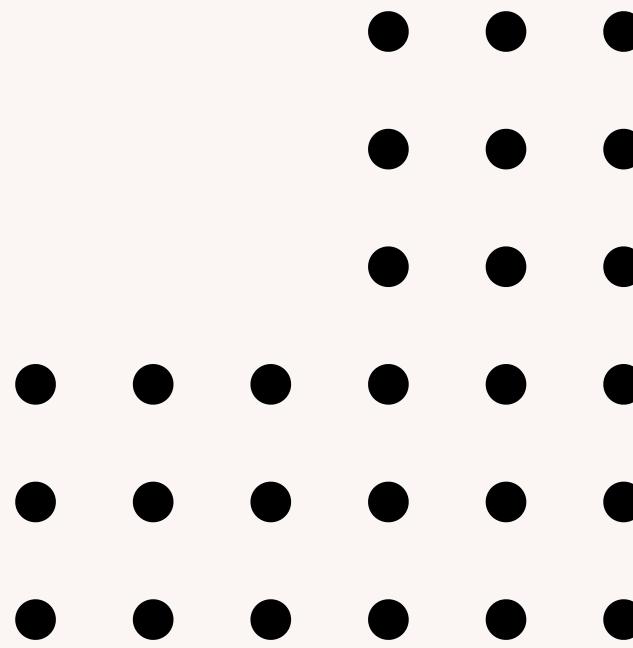
# — ATIVIDADE PRATICA – JUMP GAME JS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Jump Game</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    
    <div class="game">
        <div id="character"></div>
        <div id="block"></div>
    </div>
    <p>Pontuação: <span id="scoreSpan">0</span></p>
    <script src="script.js"></script>
</body>
</html>
```

# — ATIVIDADE PRÁTICA – JUMP GAME JS

## Jump Game:

- <img>: Adiciona a imagem de fundo com a classe filtered-image.
- <div class="game">: Contêiner principal do jogo.
- <div id="character"></div>: Representa o personagem controlado pelo jogador.
- <div id="block"></div>: Representa o obstáculo que se move na tela.
- <p>: Exibe a pontuação atual do jogador.
- <script src="script.js"></script>: Vincula o arquivo JavaScript para adicionar interatividade.



# CSS

# ATIVIDADE PRÁTICA - JUMP GAME JS

## Jump Game:

- 1 - O CSS é usado para dar vida ao jogo, definindo o estilo e posicionamento dos elementos. Vamos adicionar a aparência para o personagem, o obstáculo, e a imagem de fundo.

```
1 * {  
2     padding: 0;  
3     margin: 0;  
4     overflow-x: hidden;  
5 }  
6  
7 body {  
8     background-color: □rgb(0, 0, 0);  
9 }  
10  
11 .filtered-image {  
12     mix-blend-mode: hard-light;  
13     width: 450px;  
14     height: 300px;  
15     position: absolute;  
16     top: 10px;  
17     left: 50%;  
18     transform: translateX(-50%);  
19 }  
20  
21 .game {  
22     width: 750px;  
23     height: 200px;  
24     border: 1px solid □white;  
25     margin: auto;  
26     position: absolute;  
27     top: 40px;  
28     left: 50%;  
29     transform: translateX(-50%);  
30 }  
31  
32 #character {  
33     width: 20px;  
34     height: 50px;  
35     background-color: ■red;  
36     position: relative;  
37     top: 150px;  
38 }  
39  
40 .animate {  
41     animation: jump 0.5s linear;  
42 }  
43  
44 @keyframes jump {  
45     0% { top: 150px; }  
46     30% { top: 100px; }  
47     70% { top: 100px; }  
48     100% { top: 150px; }  
49 }  
50  
51 #block {  
52     background-color: ■blue;  
53     width: 20px;  
54     height: 20px;  
55     position: relative;  
56     top: 130px;  
57     left: 500px;  
58     animation: block 3s infinite linear;  
59 }  
60  
61 @keyframes block {  
62     0% { left: 750px; }  
63     100% { left: -20px; }  
64 }  
65  
66 p {  
67     text-align: center;  
68     color: ■#0af7ff;  
69     font-size: 28px;  
70     font-weight: bold;  
71     position: absolute;  
72     top: 300px;  
73     left: 50%;  
74     transform: translateX(-50%);  
75 }
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 1 - Reset de Estilos Globais:

- Remove as margens e preenchimentos padrão de todos os elementos.
- overflow-x: hidden;: Evita a barra de rolagem horizontal.

## 2 - Estilo do Corpo:

- Define a cor de fundo da página como preto.

```
1 * {  
2     padding: 0;  
3     margin: 0;  
4     overflow-x: hidden;  
5 }  
6  
7 body {  
8     background-color: #rgb(0, 0, 0);  
9 }  
10 }
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 3 - Imagem de Fundo:

- .filtered-image: Classe aplicada à imagem de fundo.
- mix-blend-mode: hard-light;: Aplica um efeito de mistura de cores.
- width e height: Define as dimensões da imagem.
- position: absolute; top: 10px; left: 50%; transform: translateX(-50%);: Centraliza a imagem horizontalmente na página.

```
10  
11 .filtered-image {  
12   mix-blend-mode: hard-light;  
13   width: 450px;  
14   height: 300px;  
15   position: absolute;  
16   top: 10px;  
17   left: 50%;  
18   transform: translateX(-50%);  
19 }  
20
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 4 - Área do Jogo:

- ..game: Contêiner do jogo.
- width e height: Define o tamanho da área de jogo.
- border: Adiciona uma borda branca ao redor da área de jogo.
- margin: auto;: Centraliza o contêiner.
- position: absolute; top: 40px; left: 50%; transform: translateX(-50%);: Posiciona o contêiner no centro da página com um deslocamento vertical.

```
21 .game {  
22     width: 750px;  
23     height: 200px;  
24     border: 1px solid #white;  
25     margin: auto;  
26     position: absolute;  
27     top: 40px;  
28     left: 50%;  
29     transform: translateX(-50%);  
30 }
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 5 - Personagem:

- #character: ID aplicado ao personagem.
- width e height: Define o tamanho do personagem.
- background-color: Define a cor do personagem como vermelho.
- position: relative; top: 150px;: Posiciona o personagem na base da área de jogo.

```
32 #character {  
33   width: 20px;  
34   height: 50px;  
35   background-color: red;  
36   position: relative;  
37   top: 150px;  
38 }  
39
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 6 - Animação de Salto:

- .animate: Classe que aplica a animação de salto ao personagem.
- @keyframes jump: Define a animação de salto, movendo o personagem para cima e de volta.

```
40   .animate {  
41     animation: jump 0.5s linear;  
42   }  
43  
44   @keyframes jump {  
45     0% { top: 150px; }  
46     30% { top: 100px; }  
47     70% { top: 100px; }  
48     100% { top: 150px; }  
49   }  
50
```

# — ATIVIDADE PRÁTICA – JUMP GAME JS

## 7 - Obstáculo (Bloco):

- #block: ID aplicado ao obstáculo.
- background-color: Define a cor do obstáculo como azul.
- width e height: Define o tamanho do obstáculo.
- position: relative; top: 130px; left: 500px; Posiciona o obstáculo na área de jogo.
- animation: block 3s infinite linear;; Anima o obstáculo da direita para a esquerda continuamente

```
50  
51 #block {  
52     background-color: blue;  
53     width: 20px;  
54     height: 20px;  
55     position: relative;  
56     top: 130px;  
57     left: 500px;  
58     animation: block 3s infinite linear;  
59 }  
60  
61 @keyframes block {  
62     0% { left: 750px; }  
63     100% { left: -20px; }  
64 }
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 8 - Pontuação:

- Estiliza o parágrafo que exibe a pontuação.
- Centraliza o texto e define a cor, tamanho e peso da fonte.
- Posiciona a pontuação abaixo da área de jogo.

```
66   p {  
67     text-align: center;  
68     color: #0af7ff;  
69     font-size: 28px;  
70     font-weight: bold;  
71     position: absolute;  
72     top: 300px;  
73     left: 50%;  
74     transform: translateX(-50%);  
75   }  
76
```

**JS**

# ATIVIDADE PRÁTICA – JUMP GAME JS

## Jump Game:

1 - O JavaScript adiciona a funcionalidade de salto, detecção de colisão, controle de pontuação e lógica de fim de jogo.

```
1 const character = document.getElementById("character");
2 const block = document.getElementById("block");
3 let counter = 0;
4 let isJumping = false;
5
6 document.addEventListener("keydown", function(event) {
7   if (event.code === "Space") {
8     jump();
9   }
10});
11
12 function jump() {
13   if (!isJumping) {
14     isJumping = true;
15     character.classList.add("animate");
16     setTimeout(() => {
17       character.classList.remove("animate");
18       isJumping = false;
19     }, 500);
20   }
21 }
22
23 const checkDead = setInterval(() => {
24   const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));
25   const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));
26
27   if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {
28     block.style.animation = "none";
29     alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));
30     counter = 0;
31     block.style.animation = "block 3s infinite linear";
32   } else {
33     counter++;
34     document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);
35   }
36 }, 10);
37
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 1-Seleção de Elementos e Variáveis:

- character: Seleciona o elemento do personagem.
- block: Seleciona o elemento do obstáculo.
- counter: Armazena a pontuação do jogador.
- isJumping: Flag para indicar se o personagem está saltando, evitando múltiplos saltos simultâneos.

```
1 const character = document.getElementById("character");
2 const block = document.getElementById("block");
3 let counter = 0;
4 let isJumping = false;
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 2-Event Listener para o Salto:

- Adiciona um ouvinte de evento para detectar quando a barra de espaço é pressionada.
- Quando a barra de espaço (Space) é pressionada, chama a função jump().

```
6  document.addEventListener("keydown", function(event) {  
7      if (event.code === "Space") {  
8          jump();  
9      }  
10 });
```

# — ATIVIDADE PRATICA – JUMP GAME JS

## 3-Função de Salto:

- Verifica se o personagem já está saltando para evitar saltos consecutivos.
- Adiciona a classe animate para iniciar a animação de salto.
- Usa setTimeout para remover a classe após 500ms (duração da animação), permitindo que o personagem volte ao estado original e permita novos saltos.

```
12 function jump() {  
13     if (!isJumping) {  
14         isJumping = true;  
15         character.classList.add("animate");  
16         setTimeout(() => {  
17             character.classList.remove("animate");  
18             isJumping = false;  
19         }, 500);  
20     }  
21 }
```

# — ATIVIDADE PRÁTICA – JUMP GAME JS

## 4-Detecção de Colisão e Pontuação:

- `setInterval`: Executa a função a cada 10ms para verificar constantemente a posição do personagem e do obstáculo.
- `characterTop`: Obtém a posição vertical atual do personagem.
- `blockLeft`: Obtém a posição horizontal atual do obstáculo.

```
const checkDead = setInterval(() => {
  const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));
  const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));

  if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {
    block.style.animation = "none";
    alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));
    counter = 0;
    block.style.animation = "block 3s infinite linear";
  } else {
    counter++;
    document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);
  }
}, 10);
```

# — ATIVIDADE PRÁTICA – JUMP GAME JS

## 4-Detecção de Colisão e Pontuação:

- **Condição de Colisão:**

- $\text{blockLeft} < 20 \ \&\& \ \text{blockLeft} > 0$ : Verifica se o obstáculo está na mesma posição horizontal que o personagem.
- $\text{characterTop} \geq 130$ : Verifica se o personagem não está no ar (indicando uma colisão).

```
const checkDead = setInterval(() => {
  const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));
  const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));

  if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {
    block.style.animation = "none";
    alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));
    counter = 0;
    block.style.animation = "block 3s infinite linear";
  } else {
    counter++;
    document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);
  }
}, 10);
```

# — ATIVIDADE PRÁTICA – JUMP GAME JS

## 4-Detecção de Colisão e Pontuação:

- **Ação em Caso de Colisão:**

- Para a animação do obstáculo (`block.style.animation = "none";`).
- Exibe um alerta com a pontuação final.
- Reseta a pontuação e reinicia a animação do obstáculo.

- **Atualização da Pontuação:**

- Incrementa o contador a cada intervalo.
- Atualiza o elemento `scoreSpan` para exibir a pontuação atualizada.

```
const checkDead = setInterval(() => {
  const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));
  const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));

  if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {
    block.style.animation = "none";
    alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));
    counter = 0;
    block.style.animation = "block 3s infinite linear";
  } else {
    counter++;
    document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);
  }
}, 10);
```

# ATIVIDADE

# ATIVIDADE

**Com base no “jogo” apresentado, façam as seguintes modificações e adições:**

- Adicionar função de pausa
- Adicionar movimentação horizontal para o “player”
- Adicionar mais de um elemento de obstáculo no cenário, com variação de altura e cor