



FIAP

Faculdade Sistema de Informação (SI)

Davi Grabalos Galvão

Gustavo Medeiros de Faria Santos


Lívia Rodrigues Scoralick

Michael dos Santos Marotto

Vinicius Serrano Braz de Holanda

HAPPY GAME:

GameSwap



Fiap On

2025

FIAP

"A vida é como um jogo de xadrez: sempre existe uma próxima jogada." — **Fallout.**

"Um jogo atrasado pode ser bom, mas um jogo apressado é sempre ruim." — **Shigeru Miyamoto.**

LISTA DE FIGURAS

Figura 1 – Jira	6
Figura 2 – Melhorias na Home Page	7
Figura 3 – Limpando Código com Bootstrap	8
Figura 4 – Wireframes Atualizados	10
Figura 5 – Teste de Mudanças	10
Figura 6 – Sistema de Níveis de Fidelidade	11
Figura 7 – Sistema de Pontos	11
Figura 8 – Integração com Sistema de Pontos	11
Figura 9 – Estrutura Base do Array de Níveis	13
Figura 10 – Obter Nível Atual	13
Figura 11 – Obter Próximo Nível	13
Figura 12 – Calcular Progresso	14
Figura 13 – Listagem de Benefícios	14
Figura 14 – Atualização de Pontos	15
Figura 15 – Exibição de Níveis	15
Figura 16 – Validação de Idade	16
Figura 17 – Validação de Data de Expiração	16
Figura 18 – Loop para Níveis de Fidelidade	16
Figura 19 – Loop para Validação de Requisitos	17
Figura 20 – Seleção de Método de Pagamento	17
Figura 21 – Manipulação de Itens do Carrinho	18
Figura 22 – Validação de Login	18
Figura 23 – Controle de Modal	18
Figura 24 – Manipulação de Tabs	19
Figura 25 – Timeouts de Erros	19
Figura 26 – Tratamento de Erros	19
Figura 27 – Armazenamento de Pontos	20
Figura 28 – Cálculos de Progresso	20
Figura 29 – Limites de Níveis	20
Figura 30 – Formatação de Pontos	21
Figura 31 – Verificação de Limites	21
Figura 32 – Persistência de Dados	22
Figura 33 – Antes da Refatoração	24

Figura 34 – Após a Refatoração Nova Função Centralizada	24
Figura 35 – Event Listener Simplificado	24
Figura 36 – Função ValidateForm Simplificada	25
Figura 37 – Calculo de Pontos	27
Figura 38 – Pop Up de Pontos	27
Figura 39 – Soma de Pontos	27

LISTA DE QUADROS

Quadro 1 – Melhorias do Projeto	9
Quadro 2 – Estrutura de Níveis	28
Quadro 3 – Arquivos Criados/Modificados	30

SUMÁRIO

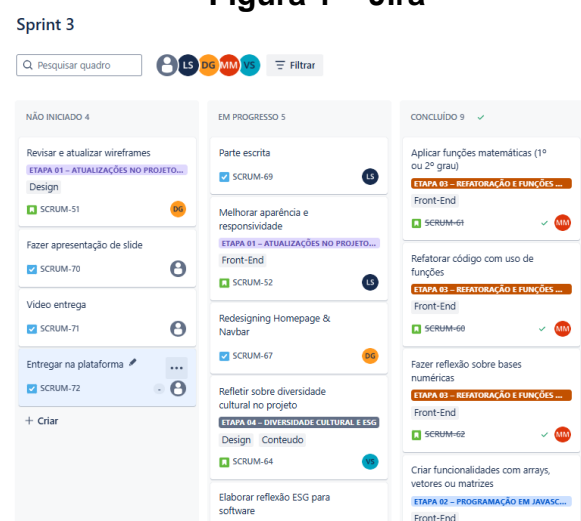
1.INTRODUÇÃO	6
2.ATUALIZAÇÕES DO PROJETO	9
2.1. Melhoria de Aparência e Responsividade.....	9
2.2. Atualização do Java Script.....	10
3. ASPECTOS DE PROGRAMAÇÃO EM JAVASCRIPT	13
3.1. Organização do código HTML	13
3.1.1. Funções de Manipulação do Array:	13
3.1.2. Uso do Array em Operações Específicas.....	14
3.1.3. Manipulação de Dados do Array:.....	14
3.1.4. Exemplos de Uso:.....	15
3.2. Estruturas de Controle (if/else)	15
3.3. Estruturas de Repetição (for/while)	16
3.4. Estruturas Switch/Case	17
3.5. Estruturas de Repetição com Arrays.....	17
3.6. Estruturas de Controle de Fluxo	18
3.7. Estruturas de Repetição com Event Listeners	18
3.8. Estruturas de Controle de Tempo	19
3.9. Estruturas de Controle de Erro	19
4. REFATORAÇÃO	20
4.1.1. Sistema de Pontos (Base Decimal)	20
4.1.2. Sistema de Níveis (Base Decimal)	20
4.1.3. Formatação de Números	21
4.1.4. Validações Numéricas	21
4.1.5. Armazenamento em LocalStorage.....	21
4.1. Boas Práticas Implementadas.....	22
4.2. Uso de Funções	23
4.3. Aplicar funções matemáticas (1º ou 2º grau)	26
4.4. Implementação Técnica	29
4.4.1. Operações Aritméticas.....	31
4.4.2. Formatação de Números	32
5. REFLEXÃO CULTURAL E ESG	34
5.1. Aspectos Culturais Afro-brasileiros e Indígenas:	34

5.2. Potenciais Culturais no GameSwap	34
5.3. Responsabilidade ESG (Ambiental, Social e Governança)	35
5.4. Conclusão.....	37
6. VÍDEO	38

1.INTRODUÇÃO

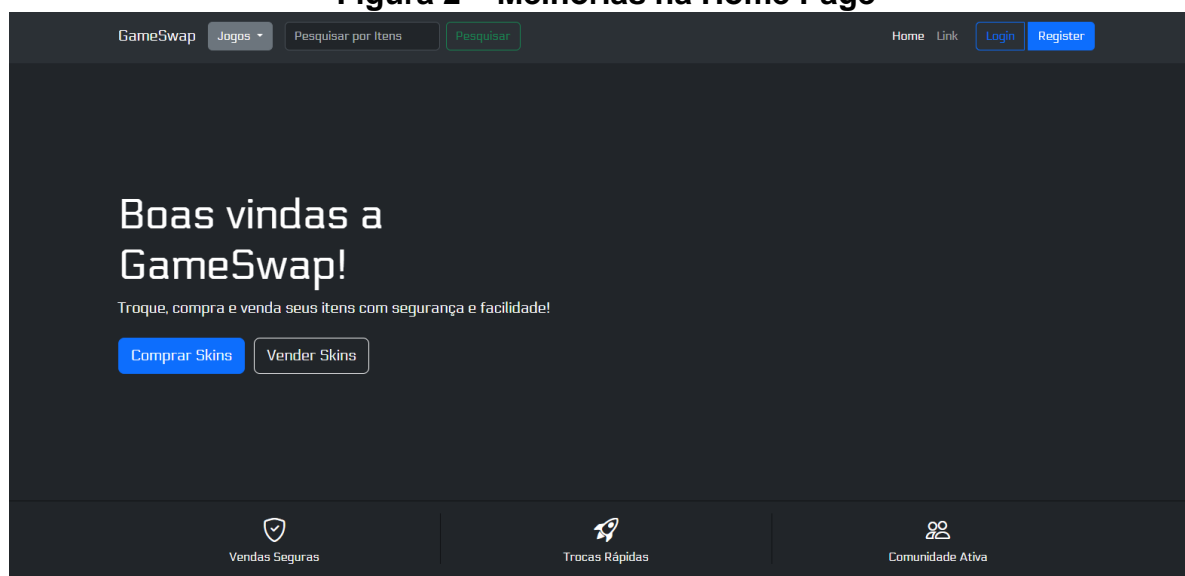
Nesta nova fase do projeto, tivemos uma surpresa importante: apresentamos nosso trabalho para a equipe da Paulo Alto. Durante essa apresentação, registramos diversas sugestões de melhorias, além de observações sobre funcionalidades implementadas por outras equipes que ainda não haviam sido incorporadas ao nosso projeto. Com base nesses feedbacks, organizamos as tarefas no Jira, incluindo também as demandas solicitadas pela FIAP, e iniciamos o processo de aprimoramento do sistema.

Figura 1 – Jira



Fonte: Elaborado pelos alunos

Inicialmente, nosso objetivo era reformular toda a aplicação utilizando o framework Bootstrap, com o intuito de modernizar o visual e torná-lo mais alinhado aos concorrentes. A seguir, apresenta-se um exemplo ilustrativo dessa proposta:

Figura 2 – Melhorias na Home Page

Fonte: Elaborado pelos alunos

Entretanto, percebemos que essa reformulação poderia comprometer a identidade visual que construímos desde o início do trabalho. Por essa razão, optamos por manter as cores e o design original em sua maioria, com exceção da página inicial (index.html), que foi a única a receber uma repaginação completa. Essa decisão foi facilitada após a codificação de 12 páginas, considerando o volume de trabalho necessário para reescrever as 28 páginas restantes (HTML, JavaScript e CSS) utilizando Bootstrap. Em reunião realizada via Google Meet, definimos em grupo que a adoção do Bootstrap seria aplicada apenas nas páginas relacionadas ao Inventário. A seguir, apresentamos um exemplo das práticas e treinamentos realizados pela equipe com o uso do Bootstrap:

Figura 3 – Limpando código com Bootstrap

```

1 <!-- BR -->
2 <body class="bg-dark text-light data-bs-theme="dark" style="font-family: 'Tekton', sans-serif;">
3   <div class="container py-5">
4     <div class="card bg-body-tertiary text-light border-0 shadow-lg rounded-4 p-4 mx-auto" style="max-width: 600px;">
5       <form id="registerForm">
6         <input type="text" class="form-control bg-dark text-light border-light" id="cpf" name="cpf" required />
7       </div>
8       <!-- Data de nascimento + aviso -->
9       <div class="mb-3">
10        <label form="birthDate" class="form-label text-info">Data de Nascimento</label>
11        <input type="date" class="form-control bg-dark text-light border-light" id="birthDate" name="birthDate" required />
12        <div id="ageWarning" class="text-warning mt-2 d-none">
13          <i class="fas fa-exclamation-triangle me-2"></i>
14          É necessário ter 18 anos ou mais para se cadastrar.
15        </div>
16      </div>
17      <!-- Senha -->
18      <div class="mb-3">
19        <label form="password" class="form-label text-info">Senha</label>
20        <div class="input-group">
21          <input type="password" class="form-control bg-dark text-light border-light" id="password" name="password" required />
22          <button type="button" class="btn btn-outline-secondary toggle-password">
23            <i class="fas fa-eye"></i>
24          </button>
25        </div>
26      <!-- Requisitos da senha -->
27      <div class="bg-dark-subtle p-3 rounded mt-2">
28        <p class="mb-1 text-info small">A senha deve conter:</p>
29        <ul class="list-unstyled small mb-0">
30          <li id="length"><i class="fas fa-times text-danger me-1"></i> Mínimo 8 caracteres</li>
31          <li id="uppercase"><i class="fas fa-times text-danger me-1"></i> Uma letra maiúscula</li>
32          <li id="lowercase"><i class="fas fa-times text-danger me-1"></i> Uma letra minúscula</li>
33          <li id="number"><i class="fas fa-times text-danger me-1"></i> Um número</li>
34          <li id="special"><i class="fas fa-times text-danger me-1"></i> Um caractere especial</li>
35        </ul>
36      </div>
37    </div>
38  </body>
39</html>
40
41 <!-- DOCTYPE HTML -->
42 <html lang="pt-BR">
43 <head>
44   <meta charset="UTF-8" />
45   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
46   <title>Dicas de Segurança - GameSwap</title>
47   <link rel="preconnect" href="https://fonts.googleapis.com" />
48   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
49   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
50   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
51   <link href="https://fonts.googleapis.com/css2?family=Tekton:wght@400;600;800&display=swap" rel="stylesheet" />
52   <link rel="stylesheet" href="css/security-tips.css" />
53 </head>
54 <body>
55   <!-- Navbar -->
56   <nav class="navbar navbar-expand-lg navbar-dark bg-dark px-3">
57     <a class="navbar-brand" href="#">
58       
59     </a>
60     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
61       <span class="navbar-toggler-icon"></span>
62     </button>
63     <div class="collapse navbar-collapse" id="navbarNav">
64       <ul class="navbar-nav ms-auto">
65         <li class="nav-item"><a class="nav-link" href="index.html">Home</a></li>
66         <li class="nav-item"><a class="nav-link" href="marketplace.html">Marketplace</a></li>
67         <li class="nav-item"><a class="nav-link" href="inventory.html">Meu Inventário</a></li>
68         <li class="nav-item"><a class="nav-link" href="profile.html">Perfil</a></li>
69       </ul>
70     </div>
71   </nav>
72
73   <!-- Main Content -->
74   <main class="container my-5">
75     <div class="security-tips-card p-4">
76       <div class="mb-5">
77         <h1 class="d-flex align-items-center gap-2">
78           <i class="fas fa-shield-alt text-regular"></i> Dicas de Segurança
79         </h1>
80       </div>
81     </div>
82   </main>
83 </body>
84</html>

```

Fonte: Teste elaborado pelos alunos

Na terceira fase do projeto Happy Game, nossa equipe também se desafiou a utilizar o GitHub como ferramenta para melhorar o desenvolvimento colaborativo. Realizamos reuniões conduzidas pelo líder Gustavo, que nos auxiliou na instalação e no envio do projeto para o repositório remoto. A adaptação ao GitHub ocorreu mais rapidamente do que o esperado, o que resultou em um aumento significativo na eficiência da execução das tarefas registradas no Jira.

2. ATUALIZAÇÕES DO PROJETO

Segue planilha resumindo as etapas dessa fase e as melhorias do grupo:

Quadro 1 – Melhorias do Projeto

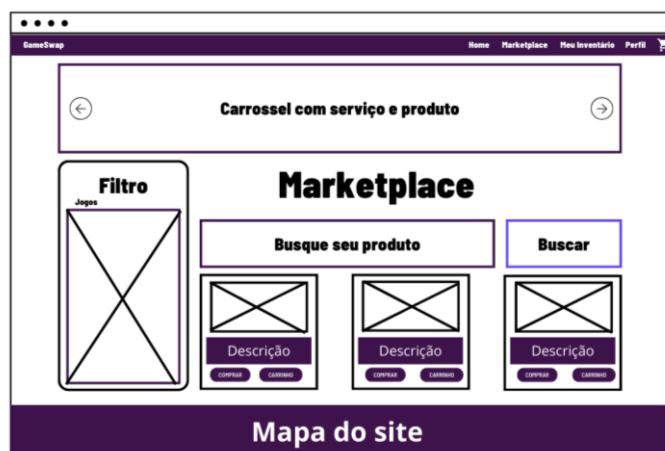
MELHORIA	FASES ANTERIORES	FASE 3
Interface e aparência	<ul style="list-style-type: none"> - Paletas não padronizadas, - Animação de background com defeito, - Footers não padronizados, 	<ul style="list-style-type: none"> - Padronização de cores e footer, - Correção de animações de background, - Mudança na wireframe, - Início de desenvolvimento de nova home, - Implementação de carrossel no marketplace e outras melhorias visuais, - Adicionamos mapa do site no fim da home, - Atualização de card de produtos
JavaScript	- Já utilizado pelo grupo.	- Revisado e alterado para melhor funcionalidade,
Bootstrap	- Página inventory.html	- Nossa equipe acha melhor para compreensão de todos continuar usando o Bootstrap em apenas uma página e css nas outras para melhor organização e separação do código.
Reflexão de funções, arrays e equações	- Já utilizado porem sem reflexão a fundo das funções.	<ul style="list-style-type: none"> - Revisados e atualizados, - Reflexão do uso e locais onde são usadas no projeto,
Práticas de ESG	- Nenhuma	<ul style="list-style-type: none"> - Implementação de VsLibras - Pesquisa de possíveis implementações para adicionar outras práticas de ESG futuramente.
Referências culturais	- Nenhuma	- Pesquisa e ideias de ações a serem tomadas pelo grupo e possíveis implementações visuais e no marketplace.

Fonte: Elaborado pelos alunos

2.1. Melhoria de Aparência e Responsividade

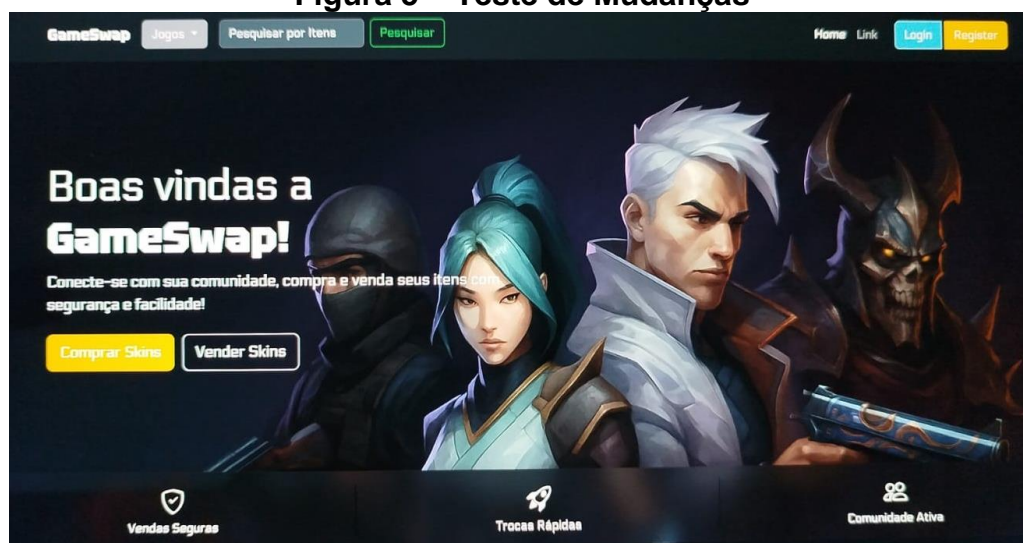
Pensando na maturidade visual do projeto e que ao entrar no site a primeira coisa a ser vista pelo cliente/usuário deve ser nossos produtos e quais os nossos serviços foi feitas novas mudanças no marketplace e testes para criar uma home mais apresentativa visualmente do nosso nicho.

Figura 4 – Wireframes atualizados
Page 1 - Marketplace



Fonte: Elaborado pelos alunos

Figura 5 – Teste de Mudanças



Fonte: Elaborado pelos alunos

2.2. Atualização do Java Script

Scripts JS existentes para melhor legibilidade e modularização:

Figura 6 – Sistema de Níveis de Fidelidade

```
// Função para obter o nível atual do usuário
export function getCurrentLevel(points) {
  for (let i = loyaltyLevels.length - 1; i >= 0; i--) {
    if (points >= loyaltyLevels[i].pointsRequired) {
      return loyaltyLevels[i];
    }
  }
  return loyaltyLevels[0];
}
```

Fonte: loyalty-levels.js e loyalty-points.js linkado a profile-points.js usado em profile.html e index.html

Figura 7 – Sistema de Pontos

```
GameSwap > GameSwapV1 > src > js > JS loyalty-points.js > ...
1 // Função para calcular pontos de fidelidade
2 export function calculateLoyaltyPoints(purchaseValue) {
3   // 1 ponto por real gasto
4   return Math.floor(purchaseValue);
5 }
```

Fonte: loyalty-points.js

Figura 8 – Integração com Sistema de Pontos

```
// Função para processar os pontos de uma compra
export function processLoyaltyPoints(purchaseValue) {
  const earnedPoints = calculateLoyaltyPoints(purchaseValue);
  const totalPoints = updateUserPoints(earnedPoints);
  displayLoyaltyPoints(totalPoints);
  return {
    earnedPoints,
    totalPoints
  };
}
```

Fonte: loyalty-points.js

No que diz respeito à organização das pastas, nosso grupo sempre seguiu as recomendações da FIAP, mantendo uma estrutura organizada e padronizada. As pastas foram separadas conforme o tipo de arquivo: HTML, JavaScript (JS), CSS, Bootstrap, além das imagens, que foram divididas em subpastas específicas para imagens gerais (img) e skins (assets).

Em relação ao uso do Bootstrap, aplicamos o framework especialmente nas páginas do inventário e na página inicial (index.html), desenvolvida pelo integrante **Davi**, garantindo maior consistência visual e responsividade nessas seções do projeto.

3. ASPECTOS DE PROGRAMAÇÃO EM JAVASCRIPT

3.1. Organização do código HTML

Nesta etapa do projeto, implementamos o uso de arrays no sistema de níveis de fidelidade:

Figura 9 – Estrutura Base do Array de Níveis

```
// Definição dos níveis de fidelidade
export const loyaltyLevels = [
  {
    level: 1,
    name: "Iniciante",
    pointsRequired: 0,
    maxPoints: 500,
    benefits: [
      "Acesso ao marketplace",
      "Sistema de pontos básico"
    ]
  },
  {
    level: 2,
    name: "Intermediário",
    pointsRequired: 500,
    maxPoints: 1000,
    benefits: [
      "Acesso ao marketplace",
      "Sistema de pontos básico",
      "Desconto de 5% em compras"
    ]
  },
  {
    level: 3,
    name: "Avançado",
    pointsRequired: 1000,
    maxPoints: 2000,
    benefits: [
      "Acesso ao marketplace",
      "Sistema de pontos básico",
      "Desconto de 10% em compras",
      "Prioridade no atendimento"
    ]
  }
];
```

Fonte: loyalty-levels.js

3.1.1. Funções de Manipulação do Array:

Figura 10 – Obter Nível Atual

```
// Função para obter o nível atual do usuário
export function getCurrentLevel(points) {
  for (let i = loyaltyLevels.length - 1; i >= 0; i--) {
    if (points >= loyaltyLevels[i].pointsRequired) {
      return loyaltyLevels[i];
    }
  }
  return loyaltyLevels[0];
}
```

Fonte: loyalty-levels.js

Figura 11 – Obter Próximo Nível

```
// Função para obter o próximo nível
export function getNextLevel(points) {
  const currentLevel = getCurrentLevel(points);
  const nextLevelIndex = loyaltyLevels.findIndex(level => level.level === currentLevel.level + 1);
  return nextLevelIndex !== -1 ? loyaltyLevels[nextLevelIndex] : null;
}
```

Fonte: loyalty-levels.js

Figura 12 – Calcular Progresso

```
// Função para calcular o progresso para o próximo nível
export function calculateLevelProgress(points) {
  const currentLevel = getCurrentLevel(points);
  const nextLevel = getNextLevel(points);

  if (!nextLevel) {
    return 100; // Já está no nível máximo
  }

  if (currentLevel.level === 1) {
    return (points / currentLevel.maxPoints) * 100;
  }

  const pointsForNextLevel = nextLevel.pointsRequired - currentLevel.pointsRequired;
  const pointsInCurrentLevel = points - currentLevel.pointsRequired;
  return (pointsInCurrentLevel / pointsForNextLevel) * 100;
}
```

Fonte: loyalty-levels.js

3.1.2. Uso do Array em Operações Específicas

Figura 13 – Listagem de Benefícios

```
// Atualiza a lista de benefícios
const benefitsList = document.getElementById('loyalty-benefits');
if (benefitsList) {
  benefitsList.innerHTML = '';
  currentLevel.benefits.forEach(benefit => {
    const li = document.createElement('li');
    li.innerHTML = `<i class="fas fa-check"></i> ${benefit}`;
    benefitsList.appendChild(li);
  });

  // Adiciona benefícios do próximo nível se houver
  if (nextLevel) {
    const nextLevelHeader = document.createElement('li');
    nextLevelHeader.className = 'next-level-benefits';
    nextLevelHeader.innerHTML = `<h4>Benefícios do próximo nível (${nextLevel.name}):</h4>`;
    benefitsList.appendChild(nextLevelHeader);

    nextLevel.benefits.forEach(benefit => {
      if (!currentLevel.benefits.includes(benefit)) {
        const li = document.createElement('li');
        li.className = 'next-benefit';
        li.innerHTML = `<i class="fas fa-lock"></i> ${benefit}`;
        benefitsList.appendChild(li);
      }
    });
  }
}
```

Fonte: profile-points.js

3.1.3. Manipulação de Dados do Array:

Figura 14 – Atualização de Pontos

```
// Função para atualizar a exibição dos pontos
function updatePointsDisplay() {
  // Recupera os pontos do localStorage
  const totalPoints = parseInt(localStorage.getItem('loyaltyPoints')) || 0;

  // Atualiza o valor total de pontos
  const pointsElement = document.getElementById('total-points');
  if (pointsElement) {
    pointsElement.textContent = totalPoints.toLocaleString('pt-BR');
  }

  // Obtém o nível atual e o próximo nível
  const currentLevel = getCurrentLevel(totalPoints);
  const nextLevel = getNextLevel(totalPoints);

  // Atualiza o nome do nível atual
  const levelNameElement = document.getElementById('current-level-name');
  if (levelNameElement) {
    levelNameElement.textContent = currentLevel.name;
  }
}
```

Fonte: profile-points.js

3.1.4. Exemplos de Uso:

Figura 15 – Exibição de Níveis

```
// Função para exibir os pontos na interface
export function displayLoyaltyPoints(points) {
  const pointsElement = document.getElementById('total-points');
  if (pointsElement) {
    pointsElement.textContent = points.toLocaleString('pt-BR');
  }
}
```

Fonte: loyalty-levels.js

3.2. Estruturas de Controle (if/else)

Nesta etapa do projeto, implementamos o uso de arrays no sistema de níveis de fidelidade:

Figura 16 – Validação de Idade

```
// Validação de idade
birthDateInput.addEventListener('change', function() {
  const birthDate = new Date(this.value);
  const age = calculateAge(birthDate);

  if (age < 18) {
    ageWarning.style.display = 'flex';
    this.setCustomValidity('É necessário ter 18 anos ou mais para se cadastrar.');
```

Fonte: register.js

Figura 17 – Validação de Data de Expiração

```
// Função para formatar a data de expiração
function formatExpirationDate(input) {
  let value = input.value.replace(/\D/g, '');

  if (value.length >= 2) {
    value = value.slice(0, 2) + '/' + value.slice(2);
  }

  input.value = value;
  updateCardPreview();
}
```

Fonte: payment.js

3.3. Estruturas de Repetição (for/while)

Figura 18 – Loop para Níveis de Fidelidade

```
// Função para obter o nível atual do usuário
export function getCurrentLevel(points) {
  for (let i = loyaltyLevels.length - 1; i >= 0; i--) {
    if (points >= loyaltyLevels[i].pointsRequired) {
      return loyaltyLevels[i];
    }
  }
  return loyaltyLevels[0];
}
```

Fonte: loyalty-levels.js

Figura 19 – Loop para Validação de Requisitos

```
Object.keys(requirements).forEach(key => {
  const element = document.getElementById(key);
  if (requirements[key]) {
    element.classList.add('valid');
    element.querySelector('i').classList.remove('fa-times');
    element.querySelector('i').classList.add('fa-check');
  } else {
    element.classList.remove('valid');
    element.querySelector('i').classList.remove('fa-check');
    element.querySelector('i').classList.add('fa-times');
  }
});

return Object.values(requirements).every(req => req);
```

Fonte: register.js

3.4. Estruturas Switch/Case

Figura 20 – Seleção de Método de Pagamento

```
// Mostrar a seção selecionada
switch(selectedMethod) {
  case 'credit-card':
    creditCardSection.style.display = 'block';
    document.getElementById('credit-card-option').classList.add('selected');
    break;
  case 'pix':
    pixSection.style.display = 'block';
    document.getElementById('pix-option').classList.add('selected');
    generatePixQRCode();
    startPixCountdown();
    break;
  case 'apple-pay':
    applePaySection.style.display = 'block';
    document.getElementById('apple-pay-option').classList.add('selected');
    break;
}
```

Fonte: payment.js

3.5. Estruturas de Repetição com Arrays

Figura 21 – Manipulação de Itens do Carrinho

```
const cartItem = document.createElement('div');
cartItem.className = 'cart-item';
cartItem.innerHTML = `
  
  <div class="cart-item-info">
    <h3>${item.name}</h3>
    <p>${item.game}</p>
    <span class="cart-item-price">R$ ${price.toFixed(2).replace('.', ',')}</span>
  </div>
  <button class="remove-item" data-index="${index}">
    <i class="fas fa-times"></i>
  </button>
`;
cartItemsContainer.appendChild(cartItem);
});
```

Fonte: profile.js

3.6. Estruturas de Controle de Fluxo

Figura 22 – Validação de Login

```
// Validações básicas
if (!email || !password) {
  alert('Por favor, preencha todos os campos!');
  return;
}
```

Fonte: login.js

Figura 23 – Controle de Modal

```
if (modalProductName) modalProductName.textContent = item.name;
if (modalProductGame) modalProductGame.textContent = item.game;
if (modalProductPrice) modalProductPrice.textContent = item.price;
if (modalProductImage) modalProductImage.src = item.image;
if (modalProductQuality) modalProductQuality.textContent = item.quality;
if (modalSellerName) modalSellerName.textContent = item.seller.name;
if (modalSellerRating) modalSellerRating.innerHTML = `<i class="fas fa-star"></i> ${item.seller.rating}`;
if (modalSellerSales) modalSellerSales.textContent = `+${item.seller.sales} vendas`;
```

Fonte: marketplace.js

3.7. Estruturas de Repetição com Event Listeners

Figura 24 – Manipulação de Tabs

```
tabButtons.forEach(button => {
  button.addEventListener('click', () => {
    // Remove active class from all buttons and contents
    tabButtons.forEach(btn => btn.classList.remove('active'));
    tabContents.forEach(content => content.classList.remove('active'));

    // Add active class to clicked button and corresponding content
    button.classList.add('active');
    const tabId = button.getAttribute('data-tab');
    document.querySelector(`#${tabId}`).classList.add('active');
  });
});
```

Fonte: lgpd-compliance.js

3.8. Estruturas de Controle de Tempo

Figura 25 – Timeouts e Intervalos

```
setInterval(() => {
  const pixels = document.querySelectorAll('.pixel');
  pixels.forEach(pixel => {
    const computedStyle = window.getComputedStyle(pixel);
    const transform = computedStyle.getPropertyValue('transform');
    if (transform.includes('matrix') && parseFloat(transform.split(',')[5]) < 0) {
      pixel.remove();
      animationContainer.appendChild(createPixel());
    }
  });
}, 1000);
```

Fonte: lgpd-compliance.js

3.9. Estruturas de Controle de Erro

Figura 26 – Tratamento de Erros

```
function showSuccess(message) {
  const successElement = document.createElement('div');
  successElement.className = 'success-message';
  successElement.textContent = message;
  document.querySelector('.payment-card').appendChild(successElement);

  setTimeout(() => {
    successElement.remove();
  }, 3000);
}
```

Fonte: payment.js

4. REFATORAÇÃO

Uso de base decimal no Projeto:

4.1.1. Sistema de Pontos (Base Decimal)

Figura 27 – Armazenamento de Pontos

```
// Função para atualizar a exibição dos pontos
function updatePointsDisplay() {
  // Recupera os pontos do localStorage
  const totalPoints = parseInt(localStorage.getItem('loyaltyPoints')) || 0;
```

Fonte: profile-points.js

- Os pontos são armazenados em base decimal
- Uso do parseInt para converter string para número decimal
- Valor padrão 0 caso não exista pontuação

Figura 28 – Cálculos de Progresso

```
// Atualiza a barra de progresso
const progressBar = document.getElementById('points-progress');
if (progressBar) {
  const progress = calculateLevelProgress(totalPoints);
  progressBar.style.width = `${progress}%`;
}
```

Fonte: profile-points.js

- Cálculos em base decimal
- Porcentagens para barra de progresso
- Operações aritméticas básicas

4.1.2. Sistema de Níveis (Base Decimal)

Figura 29 – Limites de Níveis

```
// Definição dos níveis de fidelidade
export const loyaltyLevels = [
  {
    level: 1,
    name: "Iniciante",
    pointsRequired: 0,
    maxPoints: 500,
    benefits: [
      "Acesso ao marketplace",
      "Sistema de pontos básico"
    ]
  },
  {
    level: 2,
    name: "Intermediário",
    pointsRequired: 500,
    maxPoints: 1000,
    benefits: [
      "Acesso ao marketplace",
      "Sistema de pontos básico",
      "Desconto de 5% em compras"
    ]
  },
  {
    level: 3,
    name: "Avançado",
    pointsRequired: 1000,
    maxPoints: 1500,
    benefits: [
      "Acesso ao marketplace",
      "Sistema de pontos básico",
      "Desconto de 10% em compras",
      "Prioridade no atendimento"
    ]
  }
];
```

Fonte: loyalty-levels.js

- Valores em base decimal

- Incrementos progressivos
- Validações de limites

4.1.3. Formatação de Números

Figura 30 – Formatação de Pontos

```
// Formata o valor para o código PIX (sem separadores)
const formattedValue = total.toFixed(2).replace('.', '');
pixCodeDisplay.textContent = `00020126580014BR.GOV.BCB.
PIX0136123e4567-e89b-12d3-a456-4266141740005204000053039865405${formattedValue}
5802BR5913GameSwap6008Sao Paulo62070503***6304E2CA`;
```

Fonte: payment.html

- Conversão para string com separadores
- Formatação localizada (pt-BR)
- Exibição amigável para usuário

4.1.4. Validações Numéricas

Figura 31 – Verificação de Limites

```
// Função para obter o nível atual do usuário
export function getCurrentLevel(points) {
  for (let i = loyaltyLevels.length - 1; i >= 0; i--) {
    if (points >= loyaltyLevels[i].pointsRequired) {
      return loyaltyLevels[i];
    }
  }
  return loyaltyLevels[0];
}
```

Fonte: loyalty-levels.js

- Validação de números negativos
- Verificação de limites máximos
- Tratamento de casos especiais

4.1.5. Armazenamento em LocalStorage

Figura 32 – Persistência de Dados

```

// Botão de checkout
checkoutButton.addEventListener('click', function() {
  const cart = JSON.parse(localStorage.getItem('cart')) || [];
  if (cart.length > 0) {
    // Salvar os itens do carrinho como currentPurchase
    localStorage.setItem('currentPurchase', JSON.stringify({
      items: cart
    }));
    window.location.href = '../pages/payment.html';
  } else {
    alert('Seu carrinho está vazio!');
  }
});

```

Fonte: profile.js

- Conversão para string (base decimal)
- Armazenamento como texto
- Recuperação com conversão para número

4.1. Boas Práticas Implementadas

1. Validação de Entrada
 - Verificação de números válidos
 - Tratamento de casos especiais
 - Limites de valores
2. Formatação Consistente
 - Uso de separadores de milhar
 - Formatação localizada
 - Arredondamento apropriado
3. Persistência Segura
 - Conversão para string ao armazenar
 - Conversão para número ao recuperar
 - Valores padrão seguros
4. Cálculos Precisos
 - Uso de operações aritméticas básicas
 - Tratamento de divisão por zero
 - Arredondamento quando necessário

Recomendações de Melhorias

1. Precisão Numérica

- Implementar BigInt para valores grandes
- Usar bibliotecas de precisão decimal
- Validar limites de números

2. Validações

- Adicionar mais verificações de limites
- Implementar validações de formato
- Melhorar tratamento de erros

3. Performance

- Implementar cache de cálculos
- Otimizar conversões
- Reduzir operações redundantes

4. Segurança

- Validar entradas mais rigorosamente
- Implementar verificações de integridade
- Adicionar logs de operações críticas

O projeto utiliza principalmente a base decimal para todas as operações numéricas, com foco em:

- Armazenamento de pontos
- Cálculos de progresso
- Formatação de valores
- Validações de entrada
- Persistência de dados

Não há uso direto de base binária no projeto, mas as operações são realizadas internamente pelo JavaScript em binário, com conversão automática para decimal na interface do usuário.

4.2. Uso de Funções

Nesta Fase 3, foi feita uma refatoração de código no arquivo “register.js”, segue abaixo detalhamento.

O código tinha duas partes que calculavam a idade de forma idêntica:

Figura 33 – Antes da Refatoração

```
// Primeira ocorrência - No event listener
birthDateInput.addEventListener('change', function() {
  const birthDate = new Date(this.value);
  const today = new Date();
  let age = today.getFullYear() - birthDate.getFullYear();
  const monthDiff = today.getMonth() - birthDate.getMonth();

  if (monthDiff < 0 || (monthDiff === 0 && today.getDate() < birthDate.getDate())) {
    age--;
  }
  // ... resto do código
});

// Segunda ocorrência - Na função validateForm
function validateForm() {
  const birthDate = new Date(birthDateInput.value);
  const today = new Date();
  let age = today.getFullYear() - birthDate.getFullYear();
  const monthDiff = today.getMonth() - birthDate.getMonth();

  if (monthDiff < 0 || (monthDiff === 0 && today.getDate() < birthDate.getDate())) {
    age--;
  }
  // ... resto do código
}
```

Figura 34 – Após a Refatoração Nova Função Centralizada

```
// Função reutilizável para calcular idade
function calculateAge(birthDate) {
  const today = new Date();
  let age = today.getFullYear() - birthDate.getFullYear();
  const monthDiff = today.getMonth() - birthDate.getMonth();

  if (monthDiff < 0 || (monthDiff === 0 && today.getDate() < birthDate.getDate())) {
    age--;
  }

  return age;
}
```

Figura 35 – Event Listener Simplificado

```
birthDateInput.addEventListener('change', function() {
  const birthDate = new Date(this.value);
  const age = calculateAge(birthDate); // Usando a nova função

  if (age < 18) {
    ageWarning.style.display = 'flex';
    this.setCustomValidity('É necessário ter 18 anos ou mais para se cadastrar.');
```

Figura 36 – Função validateForm Simplificada

```
function validateForm() {  
  const birthDate = new Date(birthDateInput.value);  
  const age = calculateAge(birthDate); // Usando a nova função  
  
  if (age < 18) {  
    showError('É necessário ter 18 anos ou mais para se cadastrar. Menores de idade não são permitidos na plataforma.');    return false;  
  }  
  
  return true;  
}
```

Detalhes das Mudanças:

1. Extração da Lógica
 - A lógica de cálculo de idade foi extraída para uma função separada
 - A função calculateAge recebe uma data de nascimento e retorna a idade
 - O cálculo considera o mês e o dia do nascimento para maior precisão
2. Simplificação do Código
 - Redução de 10 linhas de código duplicado
 - Melhor legibilidade e manutenibilidade
 - Código mais limpo e organizado
3. Melhorias na Manutenção
 - Se precisar alterar a lógica de cálculo de idade, só precisa mudar em um lugar
 - Menor chance de bugs por inconsistências
 - Mais fácil de testar e debugar
4. Benefícios de Performance
 - Código mais eficiente
 - Menos processamento
 - Melhor reutilização de código

Impacto nas Funcionalidades

1. Validação de Idade
 - Mantém a mesma funcionalidade
 - Mesma precisão no cálculo
 - Mesmas mensagens de erro
2. Eventos do Formulário
 - Mesmo comportamento

- Mesma validação em tempo real
 - Mesma atualização da interface
3. Experiência do Usuário
 - Nenhuma mudança perceptível
 - Mesma velocidade de resposta
 - Mesmas mensagens e feedbacks

Vantagens da Nova Implementação

1. Código Mais DRY
 - Eliminação de duplicação
 - Melhor organização
 - Mais fácil de manter
2. Melhor Testabilidade
 - Função isolada é mais fácil de testar
 - Pode ser reutilizada em outros lugares
 - Mais fácil de debugar
3. Melhor Manutenibilidade
 - Lógica centralizada
 - Menos pontos de falha
 - Mais fácil de modificar
4. Melhor Performance
 - Código mais eficiente
 - Menos processamento
 - Melhor reutilização

Esta refatoração é um exemplo clássico do princípio DRY (Don't Repeat Yourself) e melhora significativamente a qualidade do código mantendo todas as funcionalidades originais.

4.3. Aplicar funções matemáticas (1º ou 2º grau)

Segue um resumo detalhado de todas as modificações implementadas nesta branch:

Figura 37 – Calculo de Pontos

Subtotal: R\$ 2999,90

Taxa de serviço: R\$ 150,00

Desconto: R\$ 0,00

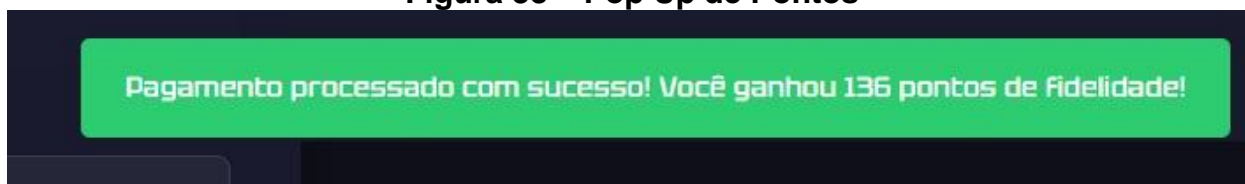
Pontos a ganhar: 3.149

Total: R\$ 3149,89

Salvar Cartão Pagar Agora

Fonte: profile.js

Figura 38 – Pop Up de Pontos



Fonte: profile.js

Figura 39 – Soma de Pontos

Meus Pontos de Fidelidade

★ Meus Pontos de Fidelidade

6.298
pontos

Nível Atual: Colecionador

3.702 pontos para Entusiasta

Seus Benefícios:

- ✓ Todos os benefícios do nível anterior
- ✓ 5% de desconto em taxas de serviço
- ✓ Prioridade no suporte

Benefícios do próximo nível (Entusiasta):

- 🔒 10% de desconto em taxas de serviço
- 🔒 Acesso antecipado a novas skins

Fonte: profile.js

Estrutura de Níveis: Criado sistema com 10 níveis de fidelidade

- Nível 1 (Iniciante): 0 a 500 pontos
- Nível 2 (Colecionador): 500+ pontos
- Nível 3 (Entusiasta): 10.000+ pontos
- Nível 4 (Veterano): 20.000+ pontos
- Nível 5 (Mestre): 35.000+ pontos
- Nível 6 (Lendário): 50.000+ pontos
- Nível 7 (Élite): 75.000+ pontos
- Nível 8 (Imperador): 100.000+ pontos
- Nível 9 (Lendário Supremo): 150.000+ pontos
- Nível 10 (GameSwap Master): 250.000+ pontos

Quadro 2 – Estrutura de Níveis

NÍVEL	PONTUAÇÃO (PONTOS)	BENEFÍCIOS
Iniciante	0 - 500	Acesso ao marketplace; Sistema de pontos básico
Colecionador	500+	5% de desconto em taxas de serviço; Prioridade no suporte
Entusiasta	10.000+	10% de desconto em taxas; Acesso antecipado a novas skins
Veterano	20.000+	15% de desconto em taxas; Cashback de 2%; Badge exclusivo no perfil

Mestre	35.000+	20% de desconto em taxas; Cashback de 3%; Acesso a eventos exclusivos; Programa de indicação com bônus
Lendário	50.000+	25% de desconto em taxas; Cashback de 4%; Suporte VIP 24/7; Acesso a skins raras exclusivas
Élite	75.000+	30% de desconto em taxas; Cashback de 5%; Concierge personalizado; Acesso a leilões exclusivos
Imperador	100.000+	35% de desconto em taxas; Cashback de 6%; Presentes mensais exclusivos; Acesso a eventos presenciais
Lendário Supremo	150.000+	40% de desconto em taxas; Cashback de 7%; Programa de recompensas personalizado; Acesso a skins únicas
GameSwap Master	250.000+	50% de desconto em taxas; Cashback de 10%; Gerente de conta dedicado; Acesso a todas as funcionalidades premium; Badge dourado exclusivo; Presentes especiais trimestrais

Fonte: Elaborado pelos alunos

4.4. Implementação Técnica

Quadro 3 – Arquivos Criados/Modificados

Arquivo	Descrição	Funções/Implementações principais
loyalty- levels.js	Implementação da estrutura de níveis	<ul style="list-style-type: none"> - getCurrentLevel: determina o nível atual do usuário - getNextLevel: identifica o próximo nível disponível - calculateLevelProgress: calcula o progresso para o próximo nível - getPointsForNextLevel: calcula pontos necessários para o próximo nível
profile- points.js	Implementação da interface do sistema de pontos	<ul style="list-style-type: none"> - Atualização dinâmica dos pontos - Cálculo e exibição do progresso - Listagem de benefícios atuais e futuros - Integração com localStorage para persistência de dados
profile.html	Adição da seção de pontos de fidelidade	<ul style="list-style-type: none"> - Implementação do card de pontos - Estrutura para exibição de benefícios - Configuração de módulos ES6

profile.css	Estilização do card de pontos	<ul style="list-style-type: none"> - Implementação da barra de progresso - Design responsivo - Cores e gradientes personalizados - Animações e transições
-------------	-------------------------------	---

Fonte: Elaborado pelos alunos

4.4.1. Operações Aritméticas

- **Adição** // Em payment.js (linha 290)

Acumulação de pontos Cálculo de progresso:

```
1 const totalPoints = currentPoints + newPoints;
```

- **Subtração** // Em loyalty-levels.js (linha 166)

Cálculo de pontos restantes para próximo nível:

```
1 return nextLevel.pointsRequired - points;
```

- **Multipliação**

Cálculo de descontos

```
1 const serviceFee = subtotal * 0.05; // Calcula 5% de taxa de serviço
```

- **Divisão**

Cálculo de progresso percentual

// Em loyalty-levels.js (linha 145)

```
1 return (points / currentLevel.maxPoints) * 100;
```

// Em loyalty-levels.js (linha 150-151)

```
1 const pointsForNextLevel = nextLevel.pointsRequired - currentLevel.pointsRequired;  
2 const pointsInCurrentLevel = points - currentLevel.pointsRequired;  
3 return (pointsInCurrentLevel / pointsForNextLevel) * 100;
```

4.4.2. Formatação de Números

- **Pontuação**

Uso de separadores de milhar

// Em profile-points.js (linha 10)

```
1 pointsElement.textContent = totalPoints.toLocaleString('pt-BR');
```

// Em payment.js (linha 294)

```
showSuccess(Pagamento processado com sucesso! Você ganhou ${pointsResult.earnedPoints.toLocaleString('pt-BR')} pontos  
de fidelidade!);
```

- **Porcentagens**

Arredondamento para 2 casas decimais

// Em payment.js (linha 420-422)

```
1 if (subtotalElement) subtotalElement.textContent = R$ ${subtotal.toFixed(2).replace('.', ',')};  
2 if (serviceFeeElement) serviceFeeElement.textContent = R$ ${serviceFee.toFixed(2).replace('.', ',')};  
3 if (totalElement) totalElement.textContent = R$ ${total.toFixed(2).replace('.', ',')};
```

5. REFLEXÃO CULTURAL E ESG

5.1. Aspectos Culturais Afro-brasileiros e Indígenas:

O Brasil é uma nação formada por múltiplas identidades, especialmente as culturas afro-brasileira e indígena, que compõem as bases da nossa herança histórica, social e espiritual. O projeto GameSwap, como proposta de plataforma inovadora de troca de itens e jogos, pode e deve ser um instrumento de valorização dessa diversidade, promovendo representatividade cultural no ambiente digital. **Sistemas de Informação Estratégicos (SIS):** A plataforma possui diferenciais competitivos claros, especialmente a especialização em itens raros, a integração futura com blockchain para segurança de transações e parcerias estratégicas com desenvolvedores de jogos independentes e grandes players do mercado.

5.2. Potenciais Culturais no GameSwap

Embora o projeto ainda não mencione diretamente a inclusão de elementos culturais afro-brasileiros ou indígenas, há diversas formas de fazê-lo:

Representação visual e estética: Incorporar grafismos indígenas e padrões africanos nos temas da interface, banners de eventos, skins ou avatares. Essas referências visuais têm forte apelo simbólico e contribuem para a valorização das culturas originárias.

Narrativas e personagens: Criar campanhas temáticas ou coleções de personagens. Essas histórias podem ser contadas por meio de enredos interativos, gamificados, que ensinem e inspirem.

Eventos culturais e educativos: Estabelecer parcerias com comunidades culturais e artistas para promover eventos especiais dentro da plataforma, como o Mês da Consciência Negra ou o Dia dos Povos Indígenas. A divulgação pode incluir conteúdos educativos sobre história, idiomas, culinária, arte, espiritualidade e música desses povos.

Jogos independentes temáticos: Abrir espaço no marketplace para jogos produzidos por desenvolvedores indígenas e negros. O incentivo à produção local não só fortalece a diversidade, mas promove inclusão econômica digital.

Mascotes representativos: Para promover diversidade e inclusão, uma ideia criativa é criar personagens ou mascotes afro-brasileiros e indígenas que apareçam em avisos no site, como banners de cookies, mensagens de boas-vindas ou durante o carregamento das páginas. Isso reforça o compromisso com a representatividade e pode tornar a experiência do usuário mais acolhedora e única.

Curation de Jogos: Destaque jogos que abordem temas relacionados à cultura afro-brasileira e indígena, ou que apresentem personagens e histórias diversas

Parcerias com Criadores: Colabore com criadores de conteúdo afro-brasileiros e indígenas para produzir vídeos, podcasts e artigos sobre jogos e cultura.

Acessibilidade (implementado): Garanta que a plataforma seja acessível para pessoas com deficiência, incluindo legendas, audiodescrição e opções de personalização.

Combate ao Preconceito: Implemente políticas de combate ao racismo, à discriminação e ao discurso de ódio na plataforma dentro do fórum, garantindo um ambiente seguro e acolhedor para todos os usuários

5.3. Responsabilidade ESG (Ambiental, Social e Governança)

O projeto GameSwap, ao evoluir para uma possível produção por uma fábrica de software, entra diretamente em um ecossistema onde práticas ESG são essenciais. O desenvolvimento de tecnologia responsável é mais do que uma tendência: é uma exigência ética e estratégica.

Ambiental – “E” de ESG

- **Infraestrutura sustentável:** Usar servidores de baixo consumo energético ou serviços com neutralidade de carbono, conforme preconizado pelo Acordo de Paris.
- **Redução de resíduos eletrônicos:** Estimular o reuso de mídia física e consoles antigos, promovendo trocas e doações em vez de descarte – prática alinhada aos ODS 12 (Consumo e Produção Responsáveis) e 13 (Ação contra a Mudança Climática).

Social – “S” de ESG

- **Inclusão digital e acessibilidade:** Criar uma plataforma acessível a pessoas com deficiência, com recursos de leitura de tela, navegação por voz e modos com contraste adaptado.
- **Diversidade na equipe e na comunidade:** Adotar políticas ativas de contratação de desenvolvedores e designers independente da sua etnia e gênero. Além disso, promover um ambiente digital seguro, com ferramentas para coibir discursos de ódio e assédio.
- **Educação e consciência crítica:** Inserir campanhas educativas sobre cidadania digital, combate ao racismo, respeito à diversidade e história cultural afro-indígena. A plataforma pode ter uma seção dedicada à “Cultura em Jogo”, com conteúdos informativos e interativos.

Governança – “G” de ESG

- **Transparência e ética digital:** Utilizar padrões como o GRI (Global Reporting Initiative) para documentar impactos sociais, ambientais e operacionais do projeto. Implementar relatórios de conformidade e boas práticas na segurança dos dados e algoritmos usados.
- **Combate ao greenwashing:** Garantir que ações ambientais e sociais do projeto sejam reais, consistentes e bem documentadas, evitando discursos vazios. Isso fortalece a confiança dos usuários e parceiros.

Uma prática crescente no e-commerce é oferecer ao cliente a opção de contribuir com um pequeno valor extra na finalização da compra para apoiar causas ambientais ou sociais, como plantio de árvores, apoio a ONGs ou compensação de carbono. Essa abordagem não só engaja o consumidor, mas também fortalece a imagem da empresa como responsável e alinhada a práticas ESG.

Essas soluções geralmente incluem dashboards para acompanhar o impacto das doações, o que pode ser exibido no próprio site para engajar ainda mais os clientes.

Sugestões de Implementação ESG para E-commerce de Jogos (HTML):

- Adicionar opção de doação no checkout para plantio de árvores ou apoio a ONGs.
- Criar badges ou conquistas digitais para clientes que contribuam com causas sustentáveis (“Eco-Gamer”, por exemplo).
- Exibir o impacto ambiental positivo das compras (ex: “Você já ajudou a plantar 10 árvores!”).
- Promover campanhas de reciclagem de eletrônicos usados. (tipo em um folder no carrossel principal)

Ao evidenciar isso no site contribuimos também para a conscientização em relação as ações a favor do ESG.

5.4. Conclusão

A adoção de práticas culturais e de ESG no projeto GameSwap não é apenas uma adição estética ou protocolar: é uma estratégia de responsabilidade, inovação e pertencimento. Ao reconhecer as culturas afro-brasileiras e indígenas e adotar compromissos reais com a sustentabilidade e a ética digital.

6. VÍDEO

Link do YouTube

FIAP