Cambios Realizados - Sistema de Totems Automáticos

Fecha

23 de Octubre, 2025

Problema Identificado

La ventana de suspensión de cuenta aparecía **ANTES** de verificar si el usuario tenía totems disponibles, incluso cuando había totems que podían ser consumidos automáticamente.

Solución Implementada

1. hooks/useTimer.ts

Cambio: Eliminada la lógica que bloqueaba automáticamente la página cuando el timer llegaba a 0.

Antes:

```
if (newTimer === 0) {
  setIsPageBlocked(true);  // ❌ Bloqueaba inmediatamente
  setShowFloatingTimer(false);

  if (options?.onTimerExpired) {
    options.onTimerExpired();
  }
}
```

Después:

```
if (newTimer === 0) {
  setShowFloatingTimer(false);

  // ✅ Solo ejecuta el callback, NO bloquea la página
  // Deja que el callback maneje la verificación de totems
  if (options?.onTimerExpired) {
    options.onTimerExpired();
  }
}
```

2. app/page.tsx

Cambio A: Definición de función global `window.openSuspensionModal`

Se agregó en el useEffect de montaje (líneas 189-205) una función global que puede ser llamada por `ContadorUsuario` cuando realmente se necesita suspender la cuenta:

```
// Definir función global para abrir el modal de suspensión
if (typeof window !== 'undefined') {
  (window as any).openSuspensionModal = () => {
    timerState.setIsPageBlocked(true);
    modalState.openModal('showZeroTimeModal');
    setActiveTab('oficina');
  };
}
```

Cambio B: Eliminación de useEffect redundante

Eliminado el useEffect que abría el modal automáticamente:

```
// ❌ ELIMINADO - Abría el modal SIN verificar totems primero
useEffect(() => {
  if (timerState.isPageBlocked && timerState.timer === 0) {
    modalState.openModal('showZeroTimeModal');
  }
}, [timerState.isPageBlocked, timerState.timer, modalState]);
```

Cambio C: Eliminación de useEffect de totems redundante

Eliminado el useEffect que intentaba consumir totems del simulador:

```
// ❌ ELIMINADO - Lógica redundante y desincronizada
useEffect(() => {
  if (timerState.isPageBlocked && simulationState.totemCount > 0) {
    if (useTotem()) {
      timerState.resetTimer();
    }
  }
}, [timerState.isPageBlocked, simulationState.totemCount]);
```

3. components/ContadorUsuario.tsx

Cambio: Agregados comentarios explicativos y manejo de errores mejorado.

El useEffect ya existía pero ahora tiene comentarios que explican claramente el flujo:

```
// Cuando el timer llega a 0, verificar PRIMERO si hay totems
useEffect(() => {
  if (timer === 0) {
    const res = await heartbeatCounter(userId);

    if (res.status === 'totem_used') {
      // ✅ Tótem consumido - reiniciar contador
      window?.YigiToast?.success?.('Tótem usado automáticamente...');
      onCounterRefreshed?.(res.counterExpiresAt);
    } else if (res.status === 'suspended') {
      // ❌ No hay totems - AHORA SÍ mostrar modal
      window?.openSuspensionModal?.();
    }
  }
}, [timer, userId]);
```

Flujo Correcto Implementado

Cuando el timer llega a 0:

1. **useTimer.ts**: Timer llega a 0 → Solo oculta el floating timer, NO bloquea la página
2. **ContadorUsuario.tsx**: Detecta timer = 0 → Llama a `heartbeatCounter(userId)`
3. **app/actions/counter.ts**:
   - ✅ Si hay totems disponibles (cantidad > 0):
     - Consume 1 tótem automáticamente
     - Reinicia `counterExpiresAt` con el tiempo completo del rango
     - Retorna `status: 'totem_used'`
   - ❌ Si NO hay totems (cantidad = 0):
     - Suspende la cuenta (`isSuspended = true`)
     - Retorna `status: 'suspended'`
1. **ContadorUsuario.tsx** (continuación):
   - Si `status === 'totem_used'` :
     - Muestra notificación de éxito
     - Reinicia el timer visual
     - **NO muestra el modal de suspensión**
   - Si `status === 'suspended'` :
     - Llama a `window.openSuspensionModal()`
1. **page.tsx**:
   - `window.openSuspensionModal()` bloquea la página y muestra el modal

Resultado

✅ **Los totems se verifican y consumen ANTES de mostrar el modal de suspensión**

✅ **El modal solo aparece cuando realmente no hay totems disponibles**

✅ **La lógica está sincronizada con el servidor (base de datos real)**

Archivos Modificados

1. `/hooks/useTimer.ts` - Líneas 83-93
2. `/app/page.tsx` - Líneas 189-205, 369-370, 379-380
3. `/components/ContadorUsuario.tsx` - Líneas 36-66

Pruebas Recomendadas

1. ✓ Usuario con totems (cantidad > 0) → Timer llega a 0 → Tótem se consume automáticamente
2. ✓ Usuario sin totems (cantidad = 0) → Timer llega a 0 → Modal de suspensión aparece
3. ✓ Usuario con 1 tótem → Timer llega a 0 dos veces → Primera vez consume tótem, segunda vez suspende
4. ✓ Notificación de “Tótem usado automáticamente” se muestra correctamente