Implementación de Modal de Confirmación para Botones de Refresh

Resumen

Se ha implementado exitosamente un modal de confirmación para los dos botones de refrescar temporizador en la aplicación YigiCoin. Cuando el usuario hace clic en cualquiera de los dos botones de refresh, ahora se muestra una ventana flotante de confirmación antes de ejecutar la acción.

Ubicación de los Botones

1. Botón en ContadorUsuario.tsx

- **Ubicación**: Componente `ContadorUsuario` (líneas 115-123)
- **Componente**: `RefreshCounterButton`
- **Descripción**: Botón flotante circular verde con ícono de refresh ubicado en la sección del contador principal

2. Botón en TopNavigation.tsx

- **Ubicación**: Barra de navegación superior (líneas 345-360)
- **Descripción**: Botón flotante circular verde en la barra de navegación junto al temporizador

Archivos Modificados

1. `/components/modals/RefreshConfirmModal.tsx`

Cambios realizados:
- Actualizado el mensaje de advertencia para incluir el texto específico requerido
- El modal ahora muestra: “Este botón extiende tu contador hasta 48 horas adicionales, pero nunca puede superar el límite máximo de tu rango.”
- Mejorada la presentación visual con un borde más prominente y mejor diseño del mensaje de advertencia

Contenido del Modal:
- ✅ Puntos actuales del usuario
- ✅ Puntos requeridos para refrescar (40 puntos)
- ✅ Tiempo actual
- ✅ Tiempo a agregar (48 horas adicionales)
- ✅ Nuevo tiempo estimado
- ✅ Techo máximo del rango
- ✅ Mensaje de advertencia: “Este botón extiende tu contador hasta 48 horas adicionales, pero nunca puede superar el límite máximo de tu rango.”
- ✅ Botones “Confirmar” y “Cancelar”

2. `/components/TopNavigation.tsx`

Cambios realizados:
- Importado `RefreshConfirmModal` y `counterSecondsForRank`

- Agregados estados:
- `userRank` : para almacenar el rango del usuario
- `showRefreshModal` : para controlar la visibilidad del modal
- Actualizada función `loadUserPoints()` para cargar también el rango del usuario
- Creada función `handleRefreshButtonClick()` que muestra el modal en lugar de ejecutar directamente la acción
- Renombrada `handleRefreshTopbar()` a `handleConfirmRefresh()` que se ejecuta después de la confirmación
- Agregado componente `RefreshConfirmModal` con todas las props requeridas
- Actualizado el onClick del botón para usar `handleRefreshButtonClick`
- Agregada lógica para cerrar el modal después de una operación exitosa
- Cambiado mensaje de éxito de “Contador restablecido” a “Temporizador refrescado”

3. `/components/RefreshCounterButton.tsx`

Estado actual:
- Ya tenía implementado el uso de `RefreshConfirmModal`
- No requirió cambios adicionales
- El modal ya estaba correctamente integrado

Flujo de Funcionamiento

Flujo Completo

1. **Usuario hace clic en botón de refresh** (en ContadorUsuario o TopNavigation)
2. **Se abre el modal de confirmación** mostrando:
   - Puntos actuales y puntos después del refresh
   - Tiempo actual y tiempo a agregar (48 horas)
   - Advertencia sobre el límite del rango
3. **Usuario puede:**
   - **Cancelar**: Cierra el modal sin hacer nada
   - **Confirmar**: Ejecuta la acción de refresh
4. **Si el usuario confirma:**
   - Se muestran indicadores de carga (spinner en botones)
   - Se intenta actualizar en el servidor
   - Si falla el servidor, usa modo demo (localStorage)
   - Se deducen 40 puntos
   - Se agrega tiempo al contador (48 horas, sin superar el límite del rango)
   - Se cierra el modal
   - Se muestra mensaje de éxito: “Temporizador refrescado (-40 puntos)”
   - Se actualiza la interfaz con el nuevo tiempo y puntos

Validaciones

- ✅ El botón está deshabilitado si el usuario no tiene suficientes puntos (< 40)
- ✅ El botón está deshabilitado si el tiempo ya está en el máximo del rango
- ✅ El tiempo agregado nunca supera el límite máximo del rango del usuario
- ✅ El modal muestra información actualizada en tiempo real

Configuración del Sistema

Economy Config (`/lib/economyConfig.ts`)

```
refreshButton: {
  cost: 40,              // Costo en puntos
  timeAddedSeconds: 48,  // +48 segundos (mostrados como 48 horas en testing)
}
```

Nota sobre Testing:
- El sistema está configurado en modo de prueba con segundos en lugar de horas reales
- La función `formatSecondsAsHours()` convierte automáticamente los segundos a “horas” para la UI
- Por lo tanto, 48 segundos se muestran como “48 horas” en la interfaz

Límites por Rango (en segundos de testing)

- **Registrado**: 168 segundos (mostrado como “168 horas”)
- **Invitado**: 72 segundos
- **Miembro**: 84 segundos
- **VIP**: 96 segundos
- **Premium**: 120 segundos
- **Elite**: 168 segundos

Mensajes del Sistema

Mensajes de Éxito

- ✅ “Temporizador refrescado (-40 puntos)” - Modo servidor
- ✅ “Temporizador refrescado (-40 puntos) [Modo Demo]” - Modo localStorage

Mensajes de Error

- ❌ “Necesitas al menos 40 puntos para refrescar el contador”
- ❌ “Puntos insuficientes (necesitas 40)”
- ❌ “No se pudo refrescar el temporizador”
- ❌ “Error al refrescar temporizador”

Tooltips de los Botones

- “Refrescar temporizador (40 puntos)” - Cuando está habilitado
- “Necesitas 40 puntos para refrescar” - Sin puntos suficientes
- “Ya estás en el tiempo máximo de tu rango” - Tiempo en el límite
- “Solo puedes refrescar cuando el tiempo sea menor a 5 minutos” - TopNavigation (restricción adicional)
- “Procesando...” - Durante la operación

Estilos y Diseño

Modal

- Fondo oscuro semitransparente (backdrop)
- Ventana blanca centrada con sombra

- Diseño responsive (máximo 448px de ancho)
- Ícono azul de refresh en la parte superior
- Secciones con colores distintivos:
- Gris: Información neutral
- Rojo: Costo
- Verde: Resultado positivo
- Azul: Tiempo a agregar
- Púrpura: Límite máximo
- Amarillo: Advertencia

Botones

- Circulares (40x40px)
- Color verde esmeralda cuando están habilitados
- Gris cuando están deshabilitados
- Animación de spinner durante la carga
- Sombra sutil con color temático

Testing

Casos de Prueba Recomendados

1. ✅ Hacer clic en el botón de refresh en ContadorUsuario
2. ✅ Hacer clic en el botón de refresh en TopNavigation
3. ✅ Verificar que el modal muestre información correcta
4. ✅ Cancelar la operación y verificar que no se deducen puntos
5. ✅ Confirmar con puntos suficientes
6. ✅ Verificar que los puntos se deducen correctamente
7. ✅ Verificar que el tiempo se agrega correctamente
8. ✅ Intentar refresh sin puntos suficientes (botón deshabilitado)
9. ✅ Intentar refresh con tiempo en el máximo del rango (botón deshabilitado)
10. ✅ Verificar mensaje de éxito después de confirmar

Compatibilidad

Modos de Operación

- ✅ **Modo Servidor**: Usa la API `/api/counter/refresh` a través de server actions
- ✅ **Modo Demo**: Usa localStorage cuando el servidor no está disponible
- ✅ **Sincronización**: Ambos modos mantienen sincronizados localStorage y el estado de la UI

Navegadores

- Compatible con todos los navegadores modernos que soporten:
- React 18+
- Next.js
- CSS Grid y Flexbox
- localStorage API

Notas Técnicas

Dependencias

- `lucide-react`: Para los íconos (RefreshCcw, Clock, etc.)
- `@/lib/economyConfig`: Configuración de economía del juego
- `@/app/actions/counter`: Server actions para operaciones del contador

State Management

- Estados locales de React (useState)
- localStorage para persistencia en modo demo
- Sincronización en tiempo real mediante intervalos

Error Handling

- Try-catch en todas las operaciones críticas
- Fallback a modo demo si el servidor falla
- Mensajes de error descriptivos para el usuario
- Logs en consola para debugging

Conclusión

La implementación está completa y cumple con todos los requisitos:
- ✅ Modal de confirmación en ambos botones de refresh
- ✅ Muestra puntos actuales y requeridos
- ✅ Muestra tiempo a agregar (48 horas adicionales)
- ✅ Muestra mensaje de advertencia específico
- ✅ Botones “Confirmar” y “Cancelar”
- ✅ Mensaje de éxito después de confirmar
- ✅ Validaciones de puntos y límites
- ✅ Diseño responsive y profesional