Sistema de Totems - Implementación Completa

Resumen de Cambios

Se ha implementado el sistema completo de totems según las especificaciones proporcionadas. Este documento detalla todos los cambios realizados en la aplicación.

📋 Especificaciones Implementadas

1. Funcionamiento del Sistema de Totems

- ✅ Cuando el temporizador del usuario llega a 0, se consume automáticamente 1 totem
- ✅ Al consumirse el totem, el temporizador se reinicia con el tiempo original establecido
- ✅ Los totems se pueden obtener de dos formas: por rango y comprándolos en la tienda

2. Distribución de Totems por Rango (Acumulativo)

```
- Registrado: 0 totems
- Invitado: 0 totems
- Básico: 0 totems
- VIP: 1 totem
- Premium: 2 totems
- Elite: 2 totems
```

Sistema Acumulativo: Si un usuario tiene rango VIP (1 totem) y sube a Premium (2 totems), queda con 3 totems en total (1 que tenía + 2 nuevos).

3. Tienda de Totems

- ✅ Precio: **1000 puntos** por totem (actualizado de 1500)
- ✅ Límite máximo: **5 totems** por usuario
- ✅ Los totems se pueden comprar en la sección “Beneficios” (tienda)

4. Interfaz de Usuario

- ✅ Cantidad de totems mostrada en la barra superior de navegación
- ✅ Notificación cuando se usa un totem automáticamente

🔧 Archivos Modificados

1. lib/economyConfig.ts

Cambios:
- Precio de totem actualizado: **1500 → 1000 puntos**
- Elite totems actualizado: **4 → 2 totems**
- Agregado límite máximo: `maxTotems: 5`

```
// Línea 53
totem: 1000, // Changed from 1500 to 1000 as per requirements

// Línea 80
elite: 2, // Changed from 4 to 2 as per requirements

// Línea 84
maxTotems: 5,
```

---

2. app/actions/store.ts

Cambios:

- Implementado límite de 5 totems en la compra
- Actualizado el costo a 1000 puntos
- Mejorado el mensaje de notificación

```
export async function buyTotem(userId?: string) {
  const cost = ECONOMY.costs.totem // 1000 (updated from 1500)
  const maxTotems = ECONOMY.maxTotems // 5

  // Check if user has reached the maximum totems limit
  if (user.totems >= maxTotems) {
    throw new Error(`Ya tienes el máximo de ${maxTotems} tótems`)
  }
  // ...
}
```

---

3. app/actions/rank-up.ts

Cambios:

- Implementado sistema acumulativo de totems
- Totems se suman al cambiar de rango (no solo se garantiza el mínimo)
- Creación de notificaciones al recibir totems por cambio de rango

```
// Base totems per rank (cumulative system)
const TOTEM_BONUS_BY_RANK: Record<UserRank, number> = {
  registrado: 0,
  invitado: 0,
  basico: 0,
  vip: 1,
  premium: 2,
  elite: 2, // Changed from 4 to 2 as per requirements
}

export async function upgradeUserRank(userId: string, newRank: UserRank) {
  // Get the totem bonus for the new rank
  const newRankBonus = TOTEM_BONUS_BY_RANK[newRank] ?? 0

  // Update rank and add totems (cumulative)
  await tx.user.update({
    where: { id: userId },
    data: {
      rank: newRank,
      totems: { increment: newRankBonus }
    },
  })
  // ...
}
```

---

4. app/actions/counter.ts

Cambios:

- Actualizado TOTEM_FLOOR para Elite: **4 → 2**

```
// Línea 30
elite: 2, // Changed from 4 to 2 as per requirements
```

Funcionalidad existente (ya implementada):

- ✅ Uso automático de totems cuando el contador expira
- ✅ Creación de notificaciones cuando se usa un totem
- ✅ Sistema de suspensión si no hay totems disponibles

---

5. components/StoreSection.tsx

Cambios:

- Precio actualizado: **1,500 → 1,000 puntos**
- Agregado estado para rastrear totems del usuario
- Implementado límite visual de 5 totems
- Deshabilitado botón de compra cuando se alcanza el límite
- Actualización automática de localStorage

```
const [userTotems, setUserTotems] = useState(0)
const maxTotems = 5 // Maximum totems allowed

// Load user totems from localStorage
useEffect(() => {
  const loadTotems = () => {
    const userData = JSON.parse(localStorage.getItem('user_simulation_data') || '{}')
    setUserTotems(userData.totems || 0)
  }
  loadTotems()
  const interval = setInterval(loadTotems, 2000)
  return () => clearInterval(interval)
}, [])

// UI showing totem count and limit
<p>Tótems actuales: <b>{userTotems}/{maxTotems}</b></p>
{userTotems >= maxTotems && (
  <p className="text-amber-600">⚠️ Ya tienes el máximo de tótems</p>
)}
```

---

6. hooks/useSimulation.ts

Cambios:

- Elite totemCount: **4 → 2**
- Sistema acumulativo implementado en upgradeToRank
- Notificaciones de totems cuando se sube de rango
- Guardado de totems en localStorage

```
// Línea 456
benefits.totemCount = 2; // Piso de tótems para Elite (changed from 4 to 2)

// En upgradeToRank:
// Add totem notification if new rank grants totems
if (totemCount > 0) {
  newNotifications.push({
    id: Date.now() + 6,
    type: 'totem',
    title: 'Tótems Recibidos',
    message: `Has recibido ${totemCount} tótem(s) por ascender a ${rankData.name}`,
    // ...
  });
}

// Calculate new totem count (cumulative system)
const newTotemCount = simulationState.totemCount + totemCount;

// Guardar en localStorage
const userData = {
  // ...
  totems: newTotemCount, // NUEVO: Guardar totems (cumulative)
};
```

📊 Sistema de Notificaciones

Tipos de Notificaciones Implementadas

1. **Totem Usado Automáticamente**
   - Se crea cuando el temporizador expira y se consume un totem
   - Tipo: `totem_used`
   - Mensaje: “Tótem usado automáticamente. Contador restablecido.”

2. **Compra de Totem Exitosa**
   - Se crea al comprar un totem en la tienda
   - Tipo: `purchase_success`
   - Mensaje: “¡Tótem comprado! Ahora tienes X tótem(s) (-1000 pts)”

3. **Compra de Totem Fallida**
   - Se crea cuando falla la compra (puntos insuficientes, límite alcanzado, etc.)
   - Tipo: `purchase_failed`

4. **Totems Recibidos por Cambio de Rango**
   - Se crea cuando el usuario sube de rango y recibe totems
   - Tipo: `totem` (en simulación) o `purchase_success` (en servidor)
   - Mensaje: “Has recibido X tótem(s) por ascender a [Rango]”

5. **Suspensión por Falta de Totems**
   - Se crea cuando el contador expira y no hay totems disponibles
   - Tipo: `suspended_for_counter`

🎨 Interfaz de Usuario

1. TopNavigation - Barra Superior

- Componente `TotemsDisplay` integrado (línea 286)
- Muestra el conteo actual de totems
- Alerta visual cuando totems = 0 (fondo rojo)
- Tooltip informativo sobre la función de los totems

2. StoreSection - Tienda

- Tarjeta de “Tótem Digital 🗿”
- Precio visible: **1,000 puntos**
- Contador de totems actuales: **X/5**
- Botón deshabilitado cuando se alcanza el límite
- Mensajes de error cuando no hay suficientes puntos

3. TotemsDisplay Component (ya existente)

- Muestra visualmente los totems del usuario
- Estilos diferentes para alerta (0 totems)
- Animaciones cuando se usa un totem
- Se actualiza automáticamente desde localStorage

🔄 Flujos del Sistema

Flujo 1: Uso Automático de Totem

```
1. Usuario tiene totems > 0
2. Temporizador llega a 0
3. Sistema verifica totems disponibles
4. Consume 1 totem (totems - 1)
5. Reinicia el temporizador según el rango
6. Crea notificación de "totem usado"
7. Actualiza UI en tiempo real
```

Flujo 2: Compra de Totem

```
1. Usuario va a Beneficios → Tienda
2. Click en "Comprar tótem"
3. Sistema verifica:
   - ¿Tiene 1000 puntos? ✓
   - ¿Tiene menos de 5 totems? ✓
4. Deduce 1000 puntos
5. Agrega 1 totem
6. Crea notificación de éxito
7. Actualiza localStorage y UI
```

Flujo 3: Cambio de Rango (Acumulativo)

```
1. Usuario sube de VIP a Premium
2. Sistema calcula totems del nuevo rango: 2
3. Suma totems: totems_actuales + 2
4. Actualiza rango y totems
5. Crea notificación: "Has recibido 2 tótem(s)"
6. Guarda en base de datos y localStorage
```

Flujo 4: Suspensión por Falta de Totems

```
1. Usuario tiene totems = 0
2. Temporizador llega a 0
3. Sistema verifica totems disponibles
4. No hay totems → Suspende cuenta
5. Establece isSuspended = true
6. Crea notificación de suspensión
7. Muestra modal de cuenta suspendida
```

🔐 Validaciones Implementadas

En Compra de Totems:

- ✅ Usuario no suspendido
- ✅ Puntos suficientes (>= 1000)
- ✅ Totems < 5 (límite máximo)
- ✅ Transacción atómica con Prisma

En Uso Automático de Totems:

- ✅ Usuario no suspendido
- ✅ Contador realmente expirado
- ✅ Al menos 1 totem disponible
- ✅ Transacción atómica con Prisma
- ✅ Garantizar piso mínimo de totems por rango

En Cambio de Rango:

- ✅ Rango válido
- ✅ Usuario existe
- ✅ Sistema acumulativo correcto
- ✅ Transacción atómica con Prisma

🗄️ Base de Datos

Schema de Prisma (ya existente)

```
model User {
  id                String    @id @default(cuid())
  email             String?   @unique
  name              String?
  rank              UserRank  @default(registrado)
  points            Int       @default(0)
  totems            Int       @default(0)
  isSuspended       Boolean   @default(false)
  suspendedAt       DateTime?
  counterExpiresAt  DateTime?
  lastTotemUsedAt   DateTime?
  createdAt         DateTime  @default(now())
  updatedAt         DateTime  @updatedAt

  notifications     Notification[]
}

enum NotificationType {
  totem_used
  suspended_for_counter
  purchase_success
  purchase_failed
  // ... otros tipos
}
```

Nota: El schema ya tiene todos los campos necesarios. Las migraciones existentes son correctas y no necesitan cambios.

🧪 Cómo Probar el Sistema

Prueba 1: Compra de Totems

```
1. Iniciar sesión con usuario de prueba
2. Ir a Beneficios → Tienda
3. Verificar que muestra "Tótems actuales: X/5"
4. Click en "Comprar tótem"
5. Verificar:
   - Puntos se deducen (-1000)
   - Totems aumentan (+1)
   - Notificación de éxito
   - UI se actualiza
```

Prueba 2: Límite de 5 Totems

```
1. Comprar totems hasta tener 5
2. Intentar comprar más
3. Verificar:
   - Botón "Comprar tótem" deshabilitado
   - Mensaje: "⚠️ Ya tienes el máximo de tótems"
   - Error si se intenta desde API
```

Prueba 3: Uso Automático de Totem

```
1. Tener al menos 1 totem
2. Esperar a que el temporizador llegue a 0
3. Verificar:
   - Totem se consume automáticamente (-1)
   - Temporizador se reinicia
   - Notificación: "Tótem usado automáticamente"
   - Cuenta no se suspende
```

Prueba 4: Sistema Acumulativo de Rangos

```
1. Usuario con rango Registrado (0 totems)
2. Subir a VIP
3. Verificar: Totems = 1
4. Subir a Premium
5. Verificar: Totems = 3 (1 anterior + 2 nuevos)
6. Verificar notificación: "Has recibido 2 tótem(s)"
```

Prueba 5: Suspensión sin Totems

```
1. Usuario con 0 totems
2. Esperar a que el temporizador llegue a 0
3. Verificar:
   - Cuenta suspendida (isSuspended = true)
   - Modal de cuenta suspendida
   - Notificación de suspensión
   - No se puede acceder a la plataforma
```

📝 Notas de Implementación

1. **Sistema Dual:**
   - El sistema funciona tanto con Prisma (base de datos) como con localStorage (modo demo/simulación)
   - Las acciones del servidor (`store.ts` , `rank-up.ts` , `counter.ts`) usan Prisma
   - El hook `useSimulation.ts` usa localStorage para demostración

2. **Sincronización:**
   - Los componentes se actualizan automáticamente cada 2 segundos desde localStorage
   - Las notificaciones se muestran en tiempo real
   - El contador de la barra de navegación se mantiene sincronizado

3. **Transacciones Atómicas:**
   - Todas las operaciones críticas usan transacciones de Prisma
   - Garantiza consistencia de datos (puntos, totems, notificaciones)

4. **Manejo de Errores:**
   - Todos los errores se capturan y se muestran al usuario
   - Se crean notificaciones de error en la base de datos
   - Los toasts visuales informan al usuario en tiempo real

🚀 Próximos Pasos Recomendados

1. **Testing:**
   - Crear tests unitarios para las acciones del servidor
   - Crear tests de integración para el flujo completo
   - Probar casos edge (suspensión, límites, etc.)

2. **Optimización:**
   - Considerar usar React Query para cacheo
   - Implementar WebSockets para actualizaciones en tiempo real
   - Optimizar consultas de Prisma con índices

3. **Monitoreo:**
   - Agregar logs para uso de totems
   - Dashboard de administración para ver estadísticas
   - Alertas cuando usuarios se quedan sin totems

4. **Mejoras UX:**
   - Animaciones más elaboradas al usar totem
   - Tutorial interactivo para nuevos usuarios
   - Predicción de cuándo se agotarán los totems

✅ Checklist de Implementación

- [x] Actualizar precio de totems (1500 → 1000)
- [x] Actualizar totems de Elite (4 → 2)
- [x] Implementar límite de 5 totems

- [x] Sistema acumulativo de totems por rango
- [x] Compra de totems en la tienda
- [x] Uso automático de totems al expirar contador
- [x] Notificaciones cuando se usa un totem
- [x] Display de totems en la barra de navegación
- [x] Validaciones de seguridad (puntos, límites)
- [x] Sincronización con localStorage
- [x] Documentación completa

📞 Soporte

Si encuentras algún problema o tienes preguntas sobre la implementación:

1. Revisar este documento primero
2. Verificar los logs de la consola
3. Revisar las notificaciones en la base de datos
4. Consultar el código en los archivos modificados listados arriba

---

Fecha de Implementación: 23 de Octubre, 2025
Versión: 1.0
Estado: ✅ Completado