YigiCoin Project Analysis Document

Generated: October 22, 2025
Purpose: Map existing components, server actions, and required implementations for Features A-H

📋 Table of Contents

---

1. Project Overview

Technology Stack

- **Framework:** Next.js 14.0.4 (App Router)
- **Language:** TypeScript 5
- **Database:** PostgreSQL with Prisma ORM 5.22.0
- **Styling:** Tailwind CSS 3.4.18
- **Icons:** RemixIcon (via lucide-react 0.263.1)
- **Node Version:** >=22 <23

Project Structure

```
yigicoin-project/
├── app/
│   ├── actions/          # Server Actions
│   │   ├── counter.ts    ✅ EXISTS (heartbeatCounter, refreshCounter)
│   │   ├── rank-up.ts    ✅ EXISTS (upgradeUserRank with totem floor logic)
│   │   └── store.ts      ✅ EXISTS (buyTotem, buyAdPackage)
│   ├── api/              # API routes
│   ├── login/
│   ├── registro/
│   ├── recuperar-password/
│   ├── page.tsx          # Main app page (Client Component)
│   ├── layout.tsx        # Root layout
│   └── globals.css       # Global styles
├── components/
│   ├── modals/
│   │   ├── SuspendedAccountModal.tsx    ✅ EXISTS
│   │   ├── FloatingTimer.tsx            ✅ EXISTS
│   │   ├── AccountModal.tsx             ✅ EXISTS
│   │   └── SupportChat.tsx              ✅ EXISTS
│   ├── BeneficiosSection.tsx   ✅ EXISTS (includes Tienda tab)
│   ├── ContadorUsuario.tsx     ✅ EXISTS (Counter display)
│   ├── RefreshCounterButton.tsx ✅ EXISTS (40 pts refresh)
│   ├── StoreSection.tsx        ✅ EXISTS (Tienda with gating)
│   ├── TopNavigation.tsx       ✅ EXISTS (Navbar with totems)
│   ├── TotemsDisplay.tsx       ✅ EXISTS (Totem counter with animations)
│   ├── NavigationTabs.tsx      ✅ EXISTS
│   ├── PanelDeControl.tsx      ✅ EXISTS
│   └── PublicidadSection.tsx   ✅ EXISTS
├── hooks/
│   ├── useTimer.ts       ✅ EXISTS (Timer management with cooldown)
│   ├── useModals.ts      ✅ EXISTS
│   ├── useSimulation.ts  ✅ EXISTS
│   └── useRefresh.ts     ✅ EXISTS
├── lib/
│   ├── prisma.ts         ✅ EXISTS
│   ├── economyConfig.ts  ✅ EXISTS (All costs, caps, helper functions)
│   └── icon-map.ts       ✅ EXISTS
└── prisma/
    └── schema.prisma     ✅ EXISTS
```

---

2. Prisma Schema Analysis

✅ Enum: UserRank

```prisma
enum UserRank {
  registrado  // 0
  invitado    // 1
  basico      // 2
  vip         // 3
  premium     // 4
  elite       // 5
}
```

✅ Enum: NotificationType

```
enum NotificationType {
  totem_used
  suspended_for_counter
  purchase_success
  purchase_failed
}
```

✅ Model: User

All required fields exist:
- `id: String` (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`
- `updatedAt: DateTime`
- `notifications: Notification[]` (relation)

✅ Model: Notification

- `id: String` (cuid)
- `userId: String` (foreign key)
- `type: NotificationType`
- `payload: Json?`
- `createdAt: DateTime`

Schema Status: ✅ **COMPLETE** - All fields required for features A-H exist

---

3. Server Actions Analysis

✅ app/actions/counter.ts

Status: Fully implemented with totem floor logic

Functions:

1. **`heartbeatCounter(userId: string)`**
   - ✅ Checks if counter expired
   - ✅ Auto-uses totem if available (atomic transaction)
   - ✅ Ensures totem floor by rank before decrementing
   - ✅ Suspends user if no totems available
   - ✅ Creates notifications (totem_used / suspended_for_counter)
   - Returns: `{ status: 'ok' | 'totem_used' | 'suspended', remainingMs, counterExpiresAt?, totems? }`

2. `refreshCounter(userId: string)`
   - ✅ Costs 40 points
   - ✅ Resets counter to full duration for current rank
   - ✅ Creates purchase_success notification
   - Returns: `{ ok: true, points, counterExpiresAt } | throws Error`

Totem Floor Logic:

```
const TOTEM_FLOOR: Record<UserRank, number> = {
  registrado: 0,
  invitado: 0,
  basico: 0,
  vip: 1,        // ✅ VIP gets at least 1 totem
  premium: 2,    // ✅ Premium gets at least 2 totems
  elite: 4,      // ✅ Elite gets at least 4 totems
}
```

✅ app/actions/store.ts

Status: Fully implemented

Functions:

1. `buyTotem(userId: string)`
   - ✅ Costs 1500 points
   - ✅ Adds 1 totem
   - ✅ Checks suspension status
   - ✅ Creates notifications (purchase_success / purchase_failed)
   - Returns: `{ ok: true, points, totems } | { ok: false, error }`

2. `buyAdPackage(userId: string)`
   - ✅ Costs 700 points
   - ✅ Grants 500 ad visits
   - ✅ Creates notifications
   - Returns: `{ ok: true, visits } | { ok: false, error }`

✅ app/actions/rank-up.ts

Status: Fully implemented with totem floor enforcement

Functions:

1. `upgradeUserRank(userId: string, newRank: UserRank)`
   - ✅ Updates user rank atomically
   - ✅ Ensures totem floor via SQL:
   `UPDATE "User" SET "totems" = GREATEST("totems", floor) WHERE "id" = userId`
   - ✅ VIP→1, Premium→2, Elite→4 totems guaranteed

---

4. Component Inventory

✅ Counter Components

components/ContadorUsuario.tsx

- **Type:** Client Component
- **Props:** `timer, isPageBlocked, resetTimer, updateTimer, setShowTimerModal, setShowInfoModal, updateButtonCooldown?, isUpdateButtonDisabled?`
- **Features:**
- Displays countdown timer (D:HH:MM:SS format)
- Shows warning when timer ≤ 300s (5 min)
- Shows suspended state when timer = 0
- Buttons: “Extender Tiempo” (opens modal), “Información”
- **Status:** ✅ Exists, needs integration with heartbeatCounter on expiry

components/RefreshCounterButton.tsx

- **Type:** Client Component
- **Props:** `userId: string, onRefreshed?: (newExpiresAt) => void`
- **Features:**
- Calls `refreshCounter(userId)` server action
- Shows loading state
- Uses `window.YigiToast` for success/error messages
- Cost: 40 points
- **Status:** ✅ Fully implemented
- **Location to add:** Inside ContadorUsuario or TopNavigation

---

✅ Store/Shop Components

components/StoreSection.tsx

- **Type:** Client Component
- **Props:** `userId: string, userRank: UserRank, points: number`
- **Features:**
- ✅ Gating: Shows lock message for `registrado` rank
- ✅ Available from `invitado` rank onwards
- ✅ Two purchase cards: Tótem digital (1500 pts), Paquete anuncios (700 pts)
- ✅ Calls `buyTotem(userId)` and `buyAdPackage(userId)`
- ✅ Shows user points balance
- ✅ Toast notifications for success/error
- **Status:** ✅ Fully implemented
- **Integration:** Already embedded in BeneficiosSection (see below)

components/BeneficiosSection.tsx

- **Type:** Client Component
- **Props:** `currentRank, digitalBooks, lotteries, selectedTheme?, userPoints?, userTotems?, onPurchaseTotem?, onPurchaseAdPackage?`

- **Features:**
- ✅ Sub-tabs: Sorteos, Libros Digitales, Loterías, **Tienda** (not shown but renders)
- ✅ Uses `canAccessStore(rank)` and `canAccessLottery(rank)` from economyConfig
- ✅ Renders store items with localStorage fallback
- ⚠️ Default tab is “tienda” but sub-tab navigation doesn’t show “Tienda” button
- **Status:** ✅ Store logic exists, needs “Tienda” tab button in sub-navigation
- **Location:** Used in main page.tsx

✅ Navigation Components

components/TopNavigation.tsx

- **Type:** Client Component
- **Props:** Many (timer, rank, modals, theme, etc.)
- **Features:**
- ✅ Displays logo, timer, user balance, points, **totems** (via TotemsDisplay), rank
- ✅ Responsive design
- ✅ User menu dropdown
- ✅ Notifications badge
- **Status:** ✅ Fully functional
- **Integration:** Used in main page.tsx

components/TotemsDisplay.tsx

- **Type:** Client Component
- **Props:** `totems?: number, selectedTheme?, showLabel?, size?: 'sm'|'md'|'lg', className?, animate?: boolean`
- **Features:**
- ✅ Displays totem count with shield icon
- ✅ Red alert style when totems = 0
- ✅ Purple normal style when totems > 0
- ✅ Animation support (pulse effect)
- ✅ Loads from localStorage if not provided
- ✅ Auto-refreshes every 2 seconds
- **Status:** ✅ Fully implemented
- **Location:** Used in TopNavigation

✅ Modal Components

components/modals/SuspendedAccountModal.tsx

- **Type:** Client Component
- **Props:** `show, onClose, reactivationTimer, penaltyPrice, formatReactivationTimer, onShowPenaltyPayment`
- **Features:**
- ✅ Shows suspended account message
- ✅ Displays countdown for 48hr reactivation window

- ✅ Shows penalty amount
- ✅ “Pagar sanción” button
- ✅ Details modal for penalty breakdown
- **Status:** ✅ Fully implemented
- **Use Case:** Open when `heartbeatCounter` returns `status: 'suspended'`

components/modals/FloatingTimer.tsx

- **Type:** Client Component
- **Props:** `show, timer, onResetTimer, updateButtonCooldown?, isUpdateButtonDisabled?`
- **Features:**
- ✅ Floats at bottom-left when timer ≤ 60s
- ✅ Shows countdown and “Click para reiniciar”
- ✅ Calls `onResetTimer()` on click
- ✅ Respects cooldown state
- **Status:** ✅ Fully implemented

---

5. Hooks & Utilities

✅ hooks/useTimer.ts

- **Type:** Client hook
- **Features:**
- ✅ Manages timer, penaltyTimer, reactivationTimer
- ✅ Auto-decrement every second
- ✅ Shows floating timer when ≤60s
- ✅ Blocks page when timer = 0
- ✅ `resetTimer()` - Full reset
- ✅ `updateTimer()` - Restart with 60s cooldown
- ✅ `addTime(seconds)` - Add time to current countdown
- ✅ Format functions for D:HH:MM:SS, MM:SS
- **Status:** ✅ Fully functional
- **Note:** Currently doesn’t call heartbeatCounter on expiry (needs integration)

✅ lib/economyConfig.ts

- **Type:** Configuration file
- **Features:**
- ✅ All economy constants in one place
- ✅ `ECONOMY.costs` : refreshCounter (40), totem (1500), adPackage (700), raffles (200/800)
- ✅ `ECONOMY.counterHours` per rank
- ✅ `ECONOMY.baseTotems` per rank (VIP:1, Premium:2, Elite:4)
- ✅ `canAccessStore(rank)` - Returns true for invitado+
- ✅ `canAccessLottery(rank)` - Returns true for invitado+
- ✅ `counterMsForRank(rank)` - Returns ms duration
- ✅ `STORE_ITEMS` config
- ✅ `LOTTERY_CONFIG` (weekly: 200pts, monthly: 800pts)

- **Status:** ✅ Complete and well-structured

---

6. Styling & Toast System

✅ Tailwind CSS

- **Status:** Configured and used throughout
- **File:** `tailwind.config.js`
- **Global Styles:** `app/globals.css` (gradient background, grid overlay)

⚠️ Toast System (window.YigiToast)

- **Current Usage:**
- `window?.YigiToast?.success?.(message)`
- `window?.YigiToast?.error?.(message)`
- **Where Used:**
- RefreshCounterButton.tsx
- StoreSection.tsx
- **Status:** ⚠️ **NOT DEFINED** - Uses optional chaining (won't crash if undefined)
- **Recommendation:** Define `window.YigiToast` in `app/layout.tsx` or create a proper toast provider

Possible Implementation:

```
// In app/layout.tsx or separate toast component
useEffect(() => {
  if (typeof window !== 'undefined') {
    window.YigiToast = {
      success: (msg: string) => alert('✅ ' + msg), // Replace with proper toast lib-
rary
      error: (msg: string) => alert('❌ ' + msg),
    };
  }
}, []);
```

---

7. Feature Requirements Mapping

Feature A: Botón “Refrescar contador (40 pts)”

Requirement	Status	Location
Server action `refreshCounter(userId)`	✅ Exists	app/actions/counter.ts: 150-183
RefreshCounterButton component	✅ Exists	components/RefreshCounterButton.tsx
Integration in counter UI	⚠️ **TODO**	Need to add to ContadorUsuario.tsx or page.tsx
Toast notifications	✅ Implemented	Uses window.YigiToast
Cost: 40 points	✅ Correct	ECONOMY.costs.refreshCounter = 40
Timer reset without reload	✅ Works	onRefreshed callback updates counterExpiresAt

Integration Required:
- Add `<RefreshCounterButton userId={userId} onRefreshed={handleRefresh} />` to main counter display
- Pass userId from server component (SSR) to ContadorUsuario as prop
- Handle `onRefreshed` to update timer state

Feature B: Tótems aumentan al ascender de rango

Requirement	Status	Location
upgradeUserRank server action	✅ Exists	app/actions/rank-up.ts:19-31
Totem floor logic (VIP≥1, Premium≥2, Elite≥4)	✅ Implemented	Uses GREATEST SQL function
Atomic transaction	✅ Yes	Wrapped in prisma.$transaction
Called after rank change	⚠️ **TODO**	Need to integrate in rank upgrade flow

Integration Required:
- In your existing rank upgrade logic (likely in useSimulation.ts or page.tsx), call:

`typescript`
`await upgradeUserRank(userId, newRank)`
- Or ensure existing rank update code includes the totem floor logic

---

Feature C: Tienda visible desde Invitado

Requirement	Status	Location
StoreSection component	✅ Exists	components/StoreSection.tsx
Gating for registrado rank	✅ Implemented	Shows lock message
Available from invitado+	✅ Correct	Uses rankGte(userRank, 'invitado')
Buy Totem (1500 pts)	✅ Works	Calls buyTotem(userId)
Buy Ad Package (700 pts)	✅ Works	Calls buyAdPackage(userId)
Show in BeneficiosSection	⚠️ **PARTIAL**	Store tab exists but not in sub-nav

Integration Required:
- **Option 1:** Add "Tienda" button to sub-tab navigation in BeneficiosSection.tsx (line 680-727)
- **Option 2:** Create separate route/tab for Tienda in NavigationTabs
- Verify StoreSection receives correct userId, userRank, points from parent

Code Change Needed:

```
// In BeneficiosSection.tsx, add to sub-navigation:
<button
  onClick={() => handleSubTabChange('tienda')}
  className={`...same styles... ${activeSubTab === 'tienda' ? 'active' : ''}`}
>
  <i className="ri-store-line mr-2"></i>
  Tienda
</button>

// Add render function:
{activeSubTab === 'tienda' && <StoreSection userId={userId} userRank={userRank as User
Rank} points={userPoints} />}
```

Feature D: Countdown: auto-uso de tótem / suspensión

Requirement	Status	Location
heartbeatCounter server action	✅ Exists	app/actions/counter.ts:47-144
Auto-use totem on expiry	✅ Implemented	Atomic updateMany with totems > 0 check
Suspend if no totems	✅ Implemented	Sets isSuspended=true, creates notification
Totem floor enforcement before use	✅ Yes	Calls ensureTotemFloor()
Call on timer=0	⚠️ **TODO**	Need to integrate in useTimer or ContadorUsuario

Integration Required:
In `hooks/useTimer.ts`, when timer reaches 0:

```
// Around line 80-83
if (newTimer === 0) {
  setIsPageBlocked(true);
  setShowFloatingTimer(false);

  // ✅ ADD THIS:
  const res = await heartbeatCounter(userId);
  if (res.status === 'totem_used') {
    window?.YigiToast?.success?.('Tótem usado automáticamente');
    // Update timer with res.counterExpiresAt
    // Maybe call onTotemUsed callback?
  } else if (res.status === 'suspended') {
    // Open suspension modal
    onSuspended?.();
  }
}
```

Alternative: Call heartbeatCounter from page.tsx when timer expires, not inside hook.

Feature E: Lotería Components (Weekly/Monthly)

Requirement	Status	Location
Lottery configuration	✅ Exists	lib/economyConfig.ts:283-296
Weekly ticket (200 pts)	✅ Defined	LOTTERY_CONFIG.weekly.cost = 200
Monthly ticket (800 pts)	✅ Defined	LOTTERY_CONFIG.monthly.cost = 800
Lottery components	❌ **MISSING**	Need to create
Lottery server action	❌ **MISSING**	Need to create
Prisma model for tickets	❌ **MISSING**	Need to add

What Needs to Be Created:

1. **components/LotterySection.tsx** (client component)
- Display weekly/monthly lotteries with prices
- Gating: visible from invitado+
- Purchase buttons calling `buyLotteryTicket(userId, type)`
- Show user's ticket count

1. **app/actions/lottery.ts** (server action)
   ```
   typescript
   export async function buyLotteryTicket(userId: string, type: 'weekly' | 'monthly') {
     const cost = type === 'weekly' ? 200 : 800;
     // Deduct points, create ticket record
   }
   ```

2. **Prisma Schema Addition:**
   ```prisma
   model LotteryTicket {
   id String @id @default(cuid())
   userId String
   user User @relation(fields: [userId], references: [id])
   type LotteryType // enum: weekly, monthly
   drawDate DateTime
   createdAt DateTime @default(now())
   @@index([userId, type])
   }

enum LotteryType {
weekly
monthly
```

}
```

1. **Integration:** Add LotterySection to BeneficiosSection sub-tabs or main navigation

**Priority:** Medium (mentioned in requirements but not critical for core counter flow)

---

## Feature F: Rank Upgrade with Totem Floor

**Status:** ✅ **COMPLETE** (see Feature B)

---

## Feature G: Suspension Modal on Counter Expiry

**Status:** ✅ Component exists, needs trigger integration (see Feature D)

---

## Feature H: Visual Effects on Totem Use

| Requirement | Status | Location |
|---|---|---|
| Glow/pulse on totem icon | ⚠️ **PARTIAL** | TotemsDisplay.tsx has `animate` prop |
| Flash on counter | ❌ **TODO** | Need to add flash animation |
| Toast notification | ✅ Works | window.YigiToast |

**Implementation Required:**
1. When totem is used (heartbeatCounter returns 'totem_used'):
`tsx`
`<TotemsDisplay animate={true} />`

1. Add flash effect to counter:
```tsx
// In ContadorUsuario.tsx or useTimer
const [flash, setFlash] = useState(false);

// When totem used:
setFlash(true);
setTimeout(() => setFlash(false), 1000);

// In JSX:


{formatTimer(timer)}

```

1. Add flash animation to globals.css:
`css`
`@keyframes flash {`
```

```
    0%, 100% { opacity: 1; }
    50% { opacity: 0.3; background-color: rgba(34, 197, 94, 0.2); }
  }
  .animate-flash {
    animation: flash 0.5s ease-in-out;
  }
```

---

8. Implementation Checklist

✅ Already Implemented (No Work Needed)

- [x] Prisma schema with all required fields
- [x] UserRank enum (registrado → elite)
- [x] Server action: refreshCounter(userId) - 40 pts
- [x] Server action: buyTotem(userId) - 1500 pts
- [x] Server action: buyAdPackage(userId) - 700 pts
- [x] Server action: upgradeUserRank(userId, newRank) with totem floor
- [x] Server action: heartbeatCounter(userId) with auto-totem use
- [x] Component: RefreshCounterButton
- [x] Component: StoreSection with gating
- [x] Component: TotemsDisplay with animations
- [x] Component: SuspendedAccountModal
- [x] Component: FloatingTimer
- [x] economyConfig with all costs and helper functions
- [x] Totem floor logic (VIP≥1, Premium≥2, Elite≥4)

⚠️ Needs Integration (Connect Existing Pieces)

- [ ] **A.** Add RefreshCounterButton to counter UI in page.tsx
- [ ] **C.** Add “Tienda” tab button to BeneficiosSection sub-navigation
- [ ] **D.** Call heartbeatCounter(userId) when timer reaches 0
- [ ] **D.** Handle totem_used status → toast + animate totem + update timer
- [ ] **D.** Handle suspended status → open SuspendedAccountModal
- [ ] **H.** Add flash animation to counter on totem use
- [ ] **Toast System:** Define window.YigiToast properly (or use library)

❌ Needs Creation (New Components/Actions)

- [ ] **E.** Create LotterySection.tsx component
- [ ] **E.** Create app/actions/lottery.ts (buyLotteryTicket)
- [ ] **E.** Add LotteryTicket model to Prisma schema
- [ ] **E.** Add lottery to BeneficiosSection or main navigation
- [ ] **E.** Run `prisma migrate dev` for lottery table

🔧 Recommended Improvements

- [ ] Define window.YigiToast in app/layout.tsx or use toast library (sonner, react-hot-toast)
- [ ] Pass userId from server component (SSR) to client components

- [ ] Add TypeScript declaration for window.YigiToast:

```typescript
// globals.d.ts
interface Window {
  YigiToast?: {
    success: (message: string) => void;
    error: (message: string) => void;
  };
}
```

- [ ] Add error boundaries for client components
- [ ] Add loading states for async operations

---

9. Current Data Flow

Counter Flow (Needs Integration)

```
1. User lands on page.tsx (client component)
2. useTimer hook starts countdown
3. When timer = 0:
   a. ⚠️ **TODO:** Call heartbeatCounter(userId) server action
   b. If status='totem_used':
      - Show toast "Tótem usado"
      - Animate TotemsDisplay
      - Update timer with new counterExpiresAt
      - Flash counter
   c. If status='suspended':
      - Open SuspendedAccountModal
      - Block page

4. User clicks RefreshCounterButton:
   - Calls refreshCounter(userId)
   - Deducts 40 pts
   - Resets timer
   - Shows success toast
```

Store Flow (✅ Works)

```
1. User navigates to Beneficios → (need to add Tienda tab)
2. If rank < invitado: Show lock message
3. If rank ≥ invitado:
   - Display StoreSection with userId, userRank, points
4. User clicks "Comprar tótem":
   - Calls buyTotem(userId)
   - Deducts 1500 pts
   - Adds 1 totem
   - Shows toast
   - (Need to refresh user data)
```

Rank Upgrade Flow (Needs Integration)

```
1. User upgrades rank (via existing logic)
2. ⚠️ **TODO:** Call upgradeUserRank(userId, newRank)
3. Server action:
   - Updates rank in DB
   - Ensures totem floor (VIP≥1, Premium≥2, Elite≥4)
4. User data refreshed
5. UI shows new rank and totem count
```

---

10. How Components Are Currently Structured

Client vs Server Components

- **app/page.tsx:** ❌ Client Component (`'use client'`)
- Should be server component to fetch user data via SSR
- Or create separate server component wrapper
- **app/layout.tsx:** ✅ Server Component (default)
- **All components/*:** ❌ Client Components (most use `'use client'`)
- This is fine for interactive UI
- Pass data from server as props

How userId is Passed Around

Current Pattern (from code inspection):
- ⚠️ userId is **NOT** being passed from server
- Components use localStorage for simulation data
- Server actions expect explicit userId parameter

Recommended Pattern:

```tsx
// app/page.tsx (make it server component)
import { getServerSession } from 'next-auth'; // or your auth
import { prisma } from '@/lib/prisma';

export default async function HomePage() {
  const session = await getServerSession();
  const userId = session?.user?.id;

  const userData = await prisma.user.findUnique({
    where: { id: userId },
    select: { id: true, rank: true, points: true, totems: true, counterExpiresAt:
true }
  });

  return (
    <ClientDashboard
      userId={userData.id}
      rank={userData.rank}
      points={userData.points}
      totems={userData.totems}
      counterExpiresAt={userData.counterExpiresAt}
    />
  );
}
```

---

11. Styling Approach

Tailwind Classes

- ✅ Used extensively throughout components
- ✅ Responsive breakpoints (sm:, md:, lg:)
- ✅ Custom color palette (blue-500, purple-600, emerald-600, red-600)
- ✅ Animations: `animate-pulse`, `animate-spin`

Custom CSS (globals.css)

- ✅ Gradient background with grid overlay
- ✅ Radial glow effects
- ✅ No @apply usage (good for Tailwind v4 migration)

Icons

- ✅ RemixIcon classes (ri-star-line, ri-shield-line, etc.)
- ✅ Consistent icon usage across components

---

12. Next Steps for Implementation

Immediate (Critical Path)

1. **Define Toast System** (5 min)
   - Add YigiToast to window in layout.tsx or use library

2. **Integrate RefreshCounterButton** (10 min)
   - Add to ContadorUsuario.tsx or page.tsx
   - Pass userId prop

3. **Add Tienda Tab to BeneficiosSection** (15 min)
   - Add button to sub-navigation
   - Ensure StoreSection renders when tab is active

4. **Connect heartbeatCounter to Timer Expiry** (30 min)
   - Modify useTimer.ts or page.tsx
   - Handle totem_used and suspended statuses
   - Trigger animations and modals

Short Term (Feature Complete)

1. **Create Lottery Components** (2 hours)
   - LotterySection.tsx component
   - app/actions/lottery.ts
   - Prisma schema migration
   - Integrate in BeneficiosSection

2. **Add Visual Effects** (30 min)
   - Flash animation on counter when totem used
   - Ensure TotemsDisplay animates

3. **Pass userId from Server Component** (1 hour)
   - Refactor page.tsx to server component
   - Create ClientDashboard wrapper
   - Fetch user data via Prisma in SSR

Long Term (Polish)

1. **Error Handling**
   - Add error boundaries
   - Handle network failures gracefully

2. **Loading States**
   - Skeleton loaders for user data
   - Suspense boundaries

3. **Testing**
   - Test totem floor logic
   - Test counter expiry scenarios
   - Test rank upgrades

---

13. Critical Notes

⚠️ Authentication System

- **Current:** No auth system detected (uses localStorage simulation)
- **Recommendation:** Implement NextAuth.js or similar before production
- **Impact:** userId must come from authenticated session

⚠️ userId Parameter

- **All server actions require explicit userId**
- **Must be passed from client components**
- **Security:** Validate userId on server (ensure user owns the session)

✅ Database Schema

- **No migrations needed** (schema is complete)
- **Only addition:** LotteryTicket model for lottery feature (optional)

✅ TypeScript

- **Well-typed throughout**
- **No major type errors expected**
- **Suggestion:** Add window.YigiToast to global types

---

14. Testing Scenarios

Test Case 1: Counter at 0 with Totem

1. Set user: totems=1, counterExpiresAt=(past date)
2. Trigger heartbeatCounter (or wait for timer=0)
3. **Expected:**
   - Totem count decreases to 0
   - Counter resets to full duration
   - Toast: “Tótem usado automáticamente”
   - TotemsDisplay animates
   - Counter flashes green

Test Case 2: Counter at 0 without Totem

1. Set user: totems=0, counterExpiresAt=(past date)
2. Trigger heartbeatCounter
3. **Expected:**
   - User.isSuspended = true
   - Notification created (type: suspended_for_counter)
   - SuspendedAccountModal opens
   - Page blocked

Test Case 3: Refresh Counter Button

1. User has 50 points, rank=invitado
2. Click “Refrescar contador (40 pts)”
3. **Expected:**
   - Points decrease to 10
   - Counter resets to 72 hours (invitado duration)
   - Toast: “Contador restablecido por 40 puntos”
   - Button disables (cooldown)

Test Case 4: Buy Totem (Registrado)

1. User rank=registrado, points=2000

2. Navigate to Tienda
3. **Expected:**
   - See lock message: “Disponible desde el rango Invitado”
   - Buy buttons disabled

Test Case 5: Buy Totem (Invitado)

1. User rank=invitado, points=2000, totems=0
2. Navigate to Tienda → Click “Comprar tótem”
3. **Expected:**
   - Points decrease to 500
   - Totems increase to 1
   - Toast: “Tótem comprado (1500 pts)”
   - TotemsDisplay updates

Test Case 6: Rank Upgrade (VIP)

1. User rank=basico, totems=0
2. Upgrade to VIP
3. **Expected:**
   - rank = vip
   - totems = 1 (floor enforced)

Test Case 7: Rank Upgrade (VIP → Premium with existing totems)

1. User rank=vip, totems=5
2. Upgrade to Premium
3. **Expected:**
   - rank = premium
   - totems = 5 (not reduced, only floor of 2 ensured)

---

15. Files Requiring Modification

High Priority

File	Changes Needed	Complexity
app/page.tsx	Add RefreshCounterButton, integrate heartbeatCounter on timer=0	Medium
components/BeneficiosSection.tsx	Add “Tienda” tab button, ensure StoreSection renders	Low
hooks/useTimer.ts	Call heartbeatCounter when timer=0, handle totem_used/suspended	Medium
app/layout.tsx	Define window.YigiToast	Low
app/globals.css	Add flash animation keyframes	Low

Medium Priority (New Files)

File	Purpose	Complexity
components/LotterySection.tsx	Display lottery options, purchase tickets	Medium
app/actions/lottery.ts	Server action for buyLotteryTicket	Low
prisma/schema.prisma	Add LotteryTicket model	Low

Low Priority

File	Changes Needed	Complexity
globals.d.ts	Add window.YigiToast type declaration	Low
components/TotemsDisplay.tsx	Ensure animate prop is used correctly	Low

---

Summary

What Exists ✅

- Complete Prisma schema with all required fields
- All server actions for counter, store, and rank upgrades
- RefreshCounterButton component
- StoreSection component with gating
- TotemsDisplay with animation support
- SuspendedAccountModal
- Totem floor logic in rank upgrades and counter heartbeat
- economyConfig with all costs and helpers

What’s Missing ❌

- Lottery components and server actions
- LotteryTicket Prisma model

What Needs Integration ⚠️

- RefreshCounterButton into counter UI
- heartbeatCounter call when timer expires
- “Tienda” tab in BeneficiosSection navigation
- Visual effects (flash) on totem use
- Proper window.YigiToast definition
- userId passed from server components (SSR)

Estimated Work

- **Critical integrations:** 2-3 hours
- **Lottery feature (optional):** 3-4 hours
- **Polish & testing:** 2-3 hours
- **Total:** 7-10 hours

Document Version: 1.0
Last Updated: October 22, 2025
Author: DeepAgent Analysis System
Status: Ready for Implementation Phase