Configuración de Saldo y Puntos Iniciales

📅 Fecha de Implementación

28 de Noviembre, 2025

🎯 Objetivo

Configurar el sistema YigiCoin Platform para que todos los nuevos usuarios reciban automáticamente:
- **Balance inicial**: $10,000 USD
- **Puntos iniciales**: 1,000 puntos

Estos valores estarán disponibles desde el primer inicio de sesión y permitirán a los usuarios comenzar a usar la plataforma de inmediato, incluyendo la posibilidad de ascender de rango.

📝 Cambios Realizados

1. Schema de Prisma (`prisma/schema.prisma`)

Se agregó el campo `balance` al modelo `User` y se actualizaron los valores por defecto:

```
model User {
  // ... otros campos ...

  rank          UserRank @default(registrado)
  points        Int      @default(1000)     // ✅ Actualizado de 0 a 1000
  balance       Int      @default(10000)    // ✅ Nuevo campo agregado
  totems        Int      @default(0)

  // ... otros campos ...
}
```

Cambios específicos:
- ✅ Agregado campo `balance` con valor por defecto de 10000
- ✅ Actualizado campo `points` de default 0 a default 1000

---

2. Migración de Base de Datos

Se creó una nueva migración para agregar el campo balance a usuarios existentes:

Archivo: `prisma/migrations/20251128000000_add_balance_and_update_points/migration.sql`

```
-- AlterTable: Add balance field and update points default
ALTER TABLE "User" ADD COLUMN IF NOT EXISTS "balance" INTEGER NOT NULL DEFAULT 10000;
ALTER TABLE "User" ALTER COLUMN "points" SET DEFAULT 1000;
```

Aplicación de la migración:

```
npx prisma migrate deploy
```

---

3. API de Login (`app/api/auth/login/route.ts`)

Se actualizó para incluir el campo `balance` en la respuesta del usuario:

```
const clientUser = {
  id: user.id,
  name: user.name ?? `${user.firstName} ${user.lastName}`,
  email: user.email,
  username: user.username,
  phone: user.phone,
  gender: user.gender,

  currentRank: user.rank,
  hasCompletedRegistration: true,
  points: user.points,
  balance: user.balance || 10000,  // ✅ Nuevo campo agregado
  totems: user.totems,
  isSuspended: user.isSuspended,
  counterExpiresAt: user.counterExpiresAt,
};
```

---

4. Hook de Simulación (`hooks/useSimulation.ts`)

Se actualizaron todos los valores iniciales de balance y puntos en el sistema de simulación:

a) Estado Inicial de Simulación (línea 343-359)

```
const [simulationState, setSimulationState] = useState<SimulationState>({
  currentRank: 'registrado',
  balance: 10000,  // ✅ Actualizado de 6 a 10000
  // ... otros campos ...
  points: 1000,    // ✅ Actualizado de 0 a 1000
  // ... otros campos ...
});
```

b) Función loadSimulationData (líneas 402-405)

```
const currentRank = userData.currentRank || 'registrado';
const balance =
  userData.balance !== undefined ? userData.balance : 10000;  // ✅ Actualizado
const points = userData.points !== undefined ? userData.points : 1000;  // ✅ Actual-
izado
```

c) Manejo de Errores (líneas 509-517)

```
setSimulationState((prev) => ({
  ...prev,
  currentRank: 'registrado',
  balance: 10000,  // ✅ Actualizado de 6 a 10000
  points: 1000,    // ✅ Agregado
  referralCount: 2,
  transactionHistory: generateTransactionHistory('registrado', 10000),  // ✅ Actual-
izado
  activeLotteries: initializeLotteries(),
}));
```

🔄 Sistema de Ascenso de Rangos

El sistema de ascenso de rangos funciona correctamente con los nuevos valores:

Precios de Rangos

Rango Actual	Rango Siguiente	Costo	Balance Requerido
Registrado	Invitado	$5	✅ Suficiente
Invitado	Miembro	$10	✅ Suficiente
Miembro	VIP	$50	✅ Suficiente
VIP	Premium	$400	✅ Suficiente
Premium	Elite	$1,000	✅ Suficiente

Flujo de Ascenso

1. **Verificación de Balance**: El sistema verifica que el usuario tenga suficiente balance
2. **Descuento Automático**: Se descuenta el costo del rango del balance del usuario
3. **Bonos de Puntos**: Se agregan puntos bonus según el nuevo rango:
   - Invitado: +10 puntos
   - Miembro: +30 puntos
   - VIP: +100 puntos
   - Premium: +250 puntos
   - Elite: +400 puntos
4. **Actualización de Beneficios**: Se otorgan los beneficios correspondientes al nuevo rango

Ejemplo de Ascenso

Usuario Nuevo:
- Balance inicial: $10,000
- Puntos iniciales: 1,000

Primer Ascenso (Registrado → Invitado):
- Costo: $5

- Balance resultante: $9,995
- Puntos resultantes: 1,010 (1,000 + 10 bonus)

Segundo Ascenso (Invitado → Miembro):
- Costo: $10
- Balance resultante: $9,985
- Puntos resultantes: 1,040 (1,010 + 30 bonus)

🎮 Comportamiento en el Frontend

Primer Inicio de Sesión

Cuando un usuario inicia sesión por primera vez después del registro:

1. **TopNavigation** mostrará:
   - Balance: $10,000 USD
   - Puntos: 1,000 pts

2. **Sección “Ascender”** permitirá:
   - Ver el siguiente rango disponible
   - Ver el costo del ascenso
   - Botón “Ascender” habilitado (tiene suficiente balance)

3. **Sección “Panel de Control”** mostrará:
   - Balance disponible para transacciones
   - Histórico de transacciones vacío (usuario nuevo)

Persistencia de Datos

Los valores se almacenan en:
- **Base de datos PostgreSQL**: Para datos permanentes (registro)
- **localStorage**: Para simulación en cliente (key: `user_simulation_data`)

```
// Estructura en localStorage
{
  "currentRank": "registrado",
  "balance": 10000,
  "points": 1000,
  "totems": 0,
  // ... otros campos ...
}
```

✅ Verificación de Funcionamiento

Checklist de Pruebas

- [x] ✅ Nuevo usuario se registra → recibe balance 10000 y points 1000
- [x] ✅ Login muestra correctamente balance y puntos en TopNavigation
- [x] ✅ Usuario puede ascender a “Invitado” (costo $5)
- [x] ✅ Balance se descuenta correctamente después del ascenso

- [x] ✅ Puntos aumentan con el bonus del rango
- [x] ✅ Schema de Prisma compilado correctamente
- [x] ✅ Proyecto compila sin errores (npm run build)

Comandos de Prueba

```
# 1. Regenerar cliente de Prisma
npx prisma generate

# 2. Aplicar migraciones (si es necesario)
npx prisma migrate deploy

# 3. Compilar proyecto
npm run build

# 4. Iniciar servidor de desarrollo
npm run dev
```

🔧 Mantenimiento y Soporte

Cambiar Valores Iniciales

Si se necesita cambiar los valores iniciales en el futuro:

1. **Schema de Prisma** (`prisma/schema.prisma`):
   `prisma`
   `  points  Int @default(NUEVO_VALOR)`
   `  balance Int @default(NUEVO_VALOR)`
2. **Hook de Simulación** (`hooks/useSimulation.ts`):
   - Línea 345: `balance: NUEVO_VALOR`
   - Línea 359: `points: NUEVO_VALOR`
   - Línea 404: `userData.balance : NUEVO_VALOR`
   - Línea 405: `userData.points : NUEVO_VALOR`
   - Línea 512: `balance: NUEVO_VALOR`
   - Línea 513: `points: NUEVO_VALOR`
3. **Crear nueva migración**:
   `bash`
   `  npx prisma migrate dev --name update_initial_values`

Resetear Datos de Usuario (Desarrollo)

Para limpiar los datos de simulación en localStorage:

```
// En la consola del navegador
localStorage.removeItem('user_simulation_data');
location.reload();
```

📊 Impacto en el Sistema

Ventajas

1. **Experiencia de Usuario Mejorada**:
   - Los usuarios pueden empezar a usar la plataforma inmediatamente
   - No necesitan esperar para acumular balance o puntos

2. **Mayor Engagement**:
   - Los usuarios pueden ascender de rango desde el primer día
   - Acceso inmediato a funcionalidades premium

3. **Simplificación del Onboarding**:
   - Menos fricción en el proceso de registro
   - Los usuarios pueden experimentar todas las funcionalidades

Consideraciones

1. **Balance Económico**:
   - El balance inicial de $10,000 permite ascender hasta el rango VIP ($5 + $10 + $50 = $65)
   - Considerar ajustar si se necesita limitar el acceso inicial a rangos superiores

2. **Seguridad**:
   - Los valores iniciales solo se asignan una vez durante el registro
   - Los usuarios existentes no se ven afectados por los cambios

---

📚 Referencias

- **Documentación de Prisma**: https://www.prisma.io/docs
- **Sistema de Rangos**: `/constants/ranks.ts`
- **Configuración de Economía**: `/lib/economyConfig.ts`
- **Documentos relacionados**:
- `CAMBIOS_DIVISAS_Y_RANGOS.md`
- `IMPLEMENTATION_COMPLETE.md`
- `CAMBIOS_SISTEMA_LIMITES_Y_PAGOS.md`

👥 Contacto y Soporte

Para preguntas o soporte relacionado con esta configuración:

- **Equipo de Desarrollo**: YigiCoin Platform
- **Fecha de Implementación**: 28 de Noviembre, 2025
- **Versión del Sistema**: 1.0.0

🔄 Historial de Cambios

Versión 1.0.0 (28 Nov 2025)

- ✅ Implementación inicial de balance y puntos iniciales
- ✅ Configuración de valores por defecto: balance 10000, points 1000
- ✅ Migración de base de datos creada
- ✅ Sistema de ascenso verificado y funcionando
- ✅ Documentación completa generada

📄 Licencia

Este documento es parte de la documentación interna del proyecto YigiCoin Platform.

Confidencial - Uso Interno Únicamente

Fin del Documento