🚀 Guía de Instalación - Parte 2A

Esta guía te ayudará a configurar e instalar las nuevas funcionalidades de la Parte 2A de la Plataforma de Incentivos.

📋 Nuevas Funcionalidades

La Parte 2A agrega las siguientes características internas:

1. **Sistema de Gamificación**
   - Puntos y niveles
   - Insignias y logros
   - Tabla de clasificación
   - Historial de puntos

2. **Simulador de Pagos**
   - Calculadora interactiva
   - Comparación de estrategias
   - Visualización de cronogramas
   - Guardado de simulaciones

3. **Sistema de Notificaciones In-App**
   - Centro de notificaciones
   - Notificaciones en tiempo real
   - Filtros y gestión
   - Contador de no leídas

4. **Sistema de Emails**
   - Recordatorios de pago
   - Confirmaciones
   - Reportes semanales/mensuales
   - Notificaciones de logros

🔧 Prerequisitos

Asegúrate de tener instalado:

- Node.js >= 18.0.0
- npm >= 9.0.0
- Proyecto de Supabase configurado (Parte 1)
- Cuenta en un servicio de email (Resend, SendGrid, etc.)

📦 Instalación

1. Actualizar Dependencias

No se requieren nuevas dependencias npm para la Parte 2A. El proyecto ya incluye todas las librerías necesarias.

2. Ejecutar Migración de Base de Datos

Opción A: Desde Supabase Dashboard

1. Ve a tu proyecto en Supabase Dashboard (https://app.supabase.com)
2. Navega a **SQL Editor**
3. Abre el archivo `supabase-migrations/002_part2a_features.sql`
4. Copia todo el contenido
5. Pégalo en el SQL Editor
6. Haz clic en **Run** para ejecutar

Opción B: Usando Supabase CLI

```
# Instalar Supabase CLI si no lo tienes
npm install -g supabase

# Iniciar sesión
supabase login

# Vincular tu proyecto
supabase link --project-ref tu-project-ref

# Ejecutar migración
supabase db push
```

3. Configurar Supabase Edge Functions

Instalar Deno (requerido para Edge Functions)

```
# macOS / Linux
curl -fsSL https://deno.land/install.sh | sh

# Windows (PowerShell)
irm https://deno.land/install.ps1 | iex
```

Desplegar Edge Functions

```
# Navegar al directorio del proyecto
cd plataforma-incentivos-mvp

# Desplegar función de envío de emails
supabase functions deploy send-email

# Desplegar función de recordatorios
supabase functions deploy schedule-payment-reminders
```

4. Configurar Variables de Entorno

Actualiza tu archivo `.env` con las nuevas variables:

```
# Variables existentes de Parte 1
VITE_SUPABASE_URL=https://tu-proyecto.supabase.co
VITE_SUPABASE_ANON_KEY=tu-anon-key

# Nuevas variables para Parte 2A
VITE_APP_URL=http://localhost:3000

# Para Edge Functions (configurar en Supabase Dashboard)
# Settings > Edge Functions > Secrets
RESEND_API_KEY=tu-resend-api-key
# O si usas SendGrid:
SENDGRID_API_KEY=tu-sendgrid-api-key
```

Configurar Secrets en Supabase

1. Ve a **Settings** > **Edge Functions** en tu proyecto Supabase
2. Agrega los siguientes secrets:
   - `RESEND_API_KEY` (o `SENDGRID_API_KEY`)
   - `APP_URL` (URL de tu aplicación)

5. Configurar Servicio de Email

Opción A: Resend (Recomendado)

1. Crea una cuenta en Resend (https://resend.com)
2. Verifica tu dominio
3. Obtén tu API Key
4. Agrégala como secret en Supabase

Opción B: SendGrid

1. Crea una cuenta en SendGrid (https://sendgrid.com)
2. Verifica tu dominio
3. Crea una API Key
4. Agrégala como secret en Supabase
5. Modifica `supabase/functions/send-email/index.ts` para usar SendGrid

6. Configurar Cron Jobs

Para que los recordatorios de pago funcionen automáticamente, configura un cron job:

Opción A: Supabase Cron (Recomendado)

```
-- Ejecutar en SQL Editor de Supabase
SELECT cron.schedule(
  'payment-reminders-daily',
  '0 9 * * *', -- Todos los días a las 9 AM
  $$
  SELECT net.http_post(
    url := 'https://tu-proyecto.supabase.co/functions/v1/schedule-payment-reminders',
    headers := '{"Content-Type": "application/json", "Authorization": "Bearer ' || cur
rent_setting('app.settings.service_role_key') || '"}'::jsonb
  );
  $$
);
```

Opción B: Servicio Externo (Cron-job.org, etc.)

1. Crea una cuenta en Cron-job.org (https://cron-job.org)
2. Crea un nuevo cron job:
   - URL: `https://tu-proyecto.supabase.co/functions/v1/schedule-payment-reminders`
   - Método: POST
   - Headers: `Authorization: Bearer tu-service-role-key`
   - Frecuencia: Diaria a las 9 AM

7. Actualizar Cache de Leaderboard

Configura un cron job para actualizar la tabla de clasificación:

```
-- Ejecutar diariamente a medianoche
SELECT cron.schedule(
  'update-leaderboard-cache',
  '0 0 * * *',
  $$
  SELECT update_leaderboard_cache();
  $$
);
```

🧪 Verificar Instalación

1. Verificar Base de Datos

Ejecuta estas consultas en SQL Editor para verificar que las tablas se crearon:

```
-- Verificar tablas de gamificación
SELECT * FROM gamification_levels;
SELECT * FROM gamification_badges;

-- Verificar que los triggers funcionan
SELECT * FROM pg_trigger WHERE tgname LIKE '%gamification%';
```

2. Probar Edge Functions

```
# Probar función de emails localmente
supabase functions serve send-email

# En otra terminal, hacer una petición de prueba
curl -X POST http://localhost:54321/functions/v1/send-email \
  -H "Authorization: Bearer tu-anon-key"
```

3. Verificar en la Aplicación

1. Inicia la aplicación:

```
npm run dev
```

1. Inicia sesión como deudor

2. Verifica las nuevas secciones:
   - `/debtor/gamification` - Sistema de gamificación

- `/debtor/simulator` - Simulador de pagos
- Centro de notificaciones (icono de campana en el header)

🎨 Integración con Parte 1

Las nuevas funcionalidades se integran automáticamente con el código existente:

Actualizar Rutas

Agrega las nuevas rutas en `src/routes/AppRouter.jsx`:

```
// Importar nuevas páginas
import GamificationPage from '../pages/debtor/GamificationPage';
import SimulatorPage from '../pages/debtor/SimulatorPage';

// Agregar rutas en el router de deudor
<Route path="gamification" element={<GamificationPage />} />
<Route path="simulator" element={<SimulatorPage />} />
```

Actualizar Navegación

Agrega enlaces en el menú de navegación del deudor:

```
// En DashboardLayout.jsx o similar
const debtorMenuItems = [
  { name: 'Dashboard', path: '/debtor/dashboard', icon: Home },
  { name: 'Mis Deudas', path: '/debtor/debts', icon: CreditCard },
  { name: 'Ofertas', path: '/debtor/offers', icon: Gift },
  { name: 'Gamificación', path: '/debtor/gamification', icon: Trophy }, // NUEVO
  { name: 'Simulador', path: '/debtor/simulator', icon: Calculator }, // NUEVO
  { name: 'Mi Billetera', path: '/debtor/wallet', icon: Wallet },
];
```

Agregar Centro de Notificaciones

En el header del dashboard:

```
import NotificationCenter from '../components/notifications/NotificationCenter';
import { useNotifications } from '../hooks/gamification/useNotifications';

function DashboardHeader() {
  const [showNotifications, setShowNotifications] = useState(false);
  const { unreadCount } = useNotifications();

  return (
    <header>
      {/* ... otros elementos ... */}

      <button onClick={() => setShowNotifications(true)} className="relative">
        <Bell className="w-6 h-6" />
        {unreadCount > 0 && (
          <span className="absolute -top-1 -right-1 bg-red-500 text-white text-xs
rounded-full w-5 h-5 flex items-center justify-center">
            {unreadCount}
          </span>
        )}
      </button>

      <NotificationCenter
        isOpen={showNotifications}
        onClose={() => setShowNotifications(false)}
      />
    </header>
  );
}
```

📊 Datos de Prueba

Para probar el sistema de gamificación, puedes insertar datos de prueba:

```
-- Crear datos de gamificación para un usuario de prueba
INSERT INTO user_gamification (user_id, total_points, current_level)
VALUES ('tu-user-id', 250, 2);

-- Otorgar una insignia de prueba
INSERT INTO user_badges (user_id, badge_id)
SELECT 'tu-user-id', id
FROM gamification_badges
WHERE badge_type = 'first_payment';
```

🔍 Solución de Problemas

Error: “Tabla no existe”

Solución: Asegúrate de haber ejecutado la migración SQL completa.

```
-- Verificar que las tablas existen
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
AND table_name LIKE '%gamification%';
```

Error: “Edge Function no responde”

Solución: Verifica que los secrets estén configurados correctamente.

```
# Listar secrets configurados
supabase secrets list
```

Notificaciones no aparecen

Solución: Verifica que RLS esté configurado correctamente.

```
-- Verificar políticas de RLS
SELECT * FROM pg_policies WHERE tablename = 'notifications';
```

Emails no se envían

Solución:
1. Verifica que la API key del servicio de email sea correcta
2. Revisa los logs de la Edge Function
3. Verifica que el dominio esté verificado en el servicio de email

```
# Ver logs de Edge Function
supabase functions logs send-email
```

📚 Próximos Pasos

1. Personaliza los templates de email en `notificationService.js`
2. Ajusta los niveles y puntos según tus necesidades en la migración SQL
3. Configura las insignias personalizadas
4. Implementa gráficos en el simulador usando Chart.js o similar
5. Agrega más tipos de notificaciones según tus necesidades

🆘 Soporte

Si encuentras problemas:

1. Revisa los logs de Supabase
2. Verifica la consola del navegador
3. Consulta la documentación de Supabase Edge Functions
4. Revisa los comentarios en el código

📝 Notas Importantes

- Las Edge Functions requieren un plan de pago en Supabase para producción
- Los cron jobs tienen límites según tu plan de Supabase
- Configura límites de rate limiting para las APIs de email
- Considera implementar un sistema de caché para el leaderboard en producción

---

¡Listo! Tu plataforma ahora cuenta con todas las funcionalidades de la Parte 2A.