🚀 Guía de Integración - Sistema Zero-Touch Onboarding

📋 Índice

🔧 Configuración Inicial

1. Variables de Entorno

Agregar al archivo `.env`:

```
# Email Service (elegir uno)
SENDGRID_API_KEY=tu_api_key_sendgrid
# O
POSTMARK_API_KEY=tu_api_key_postmark

EMAIL_FROM=noreply@inmova.app
EMAIL_FROM_NAME=INMOVA

# Chatbot AI (elegir uno)
ABACUSAI_API_KEY=tu_api_key_abacusai
# O
OPENAI_API_KEY=tu_api_key_openai

# URL de la aplicación (ya existe)
NEXTAUTH_URL=https://www.inmova.app
```

2. Generar Migración de Prisma

```
cd /home/ubuntu/homming_vidaro/nextjs_space

# Generar migración para ScheduledEmail
yarn prisma migrate dev --name add_scheduled_emails

# O si prefieres solo generar el cliente
yarn prisma generate
```

🤖 Integración del Chatbot

Paso 1: Agregar al Layout Principal

Editar `app/layout.tsx` o donde tengas el layout root:

```
import { IntelligentChatbot } from '@/components/chatbot/IntelligentChatbot';

export default function RootLayout({ children }: { children: React.ReactNode }) {
  return (
    <html lang="es">
      <body>
        {children}

        {/* Chatbot flotante - disponible en toda la app */}
        <IntelligentChatbot
          position="bottom-right"
          accentColor="#667eea"
        />
      </body>
    </html>
  );
}
```

Paso 2: Configurar GPT-4 (Opcional)

Si tienes API key de OpenAI o Abacus.AI:

```
# Verificar que las variables estén configuradas
echo $OPENAI_API_KEY
echo $ABACUSAI_API_KEY
```

Si no tienes API key, el chatbot usará respuestas predefinidas (fallback).

📧 Integración de Emails

Paso 1: Configurar Proveedor de Email

Opción A: SendGrid

1. Crear cuenta en SendGrid (https://sendgrid.com/)
2. Generar API Key en Settings > API Keys
3. Agregar a `.env`: `SENDGRID_API_KEY=SG.xxx`

Opción B: Postmark

1. Crear cuenta en Postmark (https://postmarkapp.com/)
2. Generar Server Token
3. Agregar a `.env`: `POSTMARK_API_KEY=xxx`

Paso 2: Webhooks en Registro de Usuario

Editar `app/api/auth/signup/route.ts` (o donde manejes el registro):

```
import { emailScheduler } from '@/lib/email/email-scheduler';

export async function POST(request: Request) {
  // ... tu lógica de registro existente ...

  const newUser = await prisma.user.create({
    data: { /* ... */ }
  });

  // ⭐ NUEVO: Programar emails automáticos
  await emailScheduler.onUserRegistered(newUser.id, {
    businessModel: formData.businessModel
  });

  return NextResponse.json({ success: true });
}
```

Paso 3: Webhook en Primera Propiedad

Editar `app/api/edificios/route.ts` (o donde crees edificios):

```
import { emailScheduler } from '@/lib/email/email-scheduler';

export async function POST(request: Request) {
  const session = await getServerSession(authOptions);

  // Crear edificio
  const building = await prisma.building.create({ /* ... */ });

  // Verificar si es el primero
  const buildingsCount = await prisma.building.count({
    where: { userId: session.user.id }
  });

  if (buildingsCount === 1) {
    // ⭐ NUEVO: Felicitar por primer logro
    await emailScheduler.onFirstBuildingCreated(session.user.id);
  }

  return NextResponse.json(building);
}
```

Paso 4: Cron Job para Procesar Emails

Crear `app/api/cron/process-emails/route.ts`:

```
import { NextResponse } from 'next/server';
import { emailScheduler } from '@/lib/email/email-scheduler';

export const dynamic = 'force-dynamic';

export async function GET(request: Request) {
  // Verificar token de seguridad
  const authHeader = request.headers.get('authorization');
  if (authHeader !== `Bearer ${process.env.CRON_SECRET}`) {
    return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
  }

  // Procesar emails pendientes
  await emailScheduler.processPendingEmails();

  return NextResponse.json({ success: true });
}
```

Configurar cron job en Vercel/Railway/tu hosting:

```
# Cada 5 minutos
*/5 * * * * curl -H "Authorization: Bearer tu_cron_secret" https://www.inmova.app/api/
cron/process-emails
```

🎨 Integración Mobile First CSS

Paso 1: Importar CSS Global

Editar `app/globals.css`:

```
/* Tus estilos existentes */
@tailwind base;
@tailwind components;
@tailwind utilities;

/* ⭐ NUEVO: Mobile First Styles */
@import './styles/mobile-first.css';
```

O importar directamente en `app/layout.tsx`:

```
import '@/styles/mobile-first.css';
```

Paso 2: Aplicar Clases a Elementos Existentes

Formularios

Antes:

```
<input type="text" className="border rounded p-2" />
```

Después:

```
<input type="text" className="form-input" />
```

Botones

Asegurarse de que todos los botones tengan tamaño mínimo:

```
<Button className="min-h-[48px]">
  Crear Propiedad
</Button>
```

Contenedores

Usar el container system:

```
<div className="container">
  {/* Tu contenido */}
</div>
```

📱 Navegación Móvil

Paso 1: Agregar al Layout

Editar `app/layout.tsx`:

```
import { MobileNavigation } from '@/components/mobile/MobileNavigation';

export default function RootLayout({ children }: { children: React.NodeNode }) {
  return (
    <html lang="es">
      <body className="app-container">
        {/* Sidebar existente (oculto en móvil) */}
        <Sidebar className="hidden md:block" />

        <main className="main-content">
          {children}
        </main>

        {/* ⭐ NUEVO: Bottom Navigation (solo móvil) */}
        <MobileNavigation />

        <IntelligentChatbot />
      </body>
    </html>
  );
}
```

Paso 2: Ajustar Content Padding

Asegurarse de que el contenido principal tenga padding para el bottom nav:

```
.main-content {
  padding-bottom: calc(64px + env(safe-area-inset-bottom));
}
```

🧪 Testing

Test 1: Chatbot

```
# 1. Abrir la app en navegador
open http://localhost:3000/home

# 2. Verificar que aparece el botón flotante
# 3. Hacer clic y probar conversación
# 4. Probar acciones rápidas
```

Test 2: Emails

```
# 1. Registrar nuevo usuario
curl -X POST http://localhost:3000/api/auth/signup \
  -H "Content-Type: application/json" \
  -d '{"email":"test@test.com","password":"test123","name":"Test User"}'

# 2. Verificar que se creó el scheduled email
yarn prisma studio
# Ir a tabla "scheduled_emails" y verificar

# 3. Ejecutar procesamiento manual
curl -X GET http://localhost:3000/api/cron/process-emails \
  -H "Authorization: Bearer tu_cron_secret"

# 4. Verificar email en SendGrid/Postmark dashboard
```

Test 3: Mobile First

```
# 1. Abrir DevTools (F12)
# 2. Toggle device toolbar (Ctrl+Shift+M)
# 3. Seleccionar iPhone 12 Pro
# 4. Verificar:
#    - Bottom navigation visible
#    - Botones tamaño adecuado (>=48px)
#    - Inputs no hacen zoom al enfocar
#    - Formularios divididos en pasos
```

Test 4: Skeleton Screens

```
// Usar en páginas con carga de datos
import { DashboardSkeleton } from '@/components/ui/skeleton-screen';

function Dashboard() {
  const { data, isLoading } = useSWR('/api/dashboard');

  if (isLoading) {
    return <DashboardSkeleton />;
  }

  return <DashboardContent data={data} />;
}
```

✅ Checklist de Implementación

Configuración

- [ ] Variables de entorno configuradas (.env)
- [ ] Migración de Prisma ejecutada
- [ ] Proveedor de email configurado (SendGrid o Postmark)
- [ ] API key de GPT-4 configurada (opcional)

Integración Chatbot

- [ ] Chatbot agregado al layout root
- [ ] API route `/api/chatbot` funcionando
- [ ] Respuestas predefinidas configuradas
- [ ] Interacciones guardadas en DB

Integración Emails

- [ ] EmailService configurado
- [ ] Webhook en registro de usuario
- [ ] Webhook en primera propiedad
- [ ] Cron job configurado para procesamiento
- [ ] Templates de email personalizados

Mobile First

- [ ] CSS Mobile First importado
- [ ] Bottom navigation agregada
- [ ] Formularios optimizados para móvil
- [ ] Touch targets >=48px
- [ ] Safe areas configuradas (iOS)
- [ ] Skeleton screens implementados

Testing

- [ ] Chatbot responde correctamente
- [ ] Emails se envían automáticamente
- [ ] Navegación móvil funciona
- [ ] Responsive en todos los breakpoints
- [ ] Performance aceptable (<3seg TTI)

📊 Métricas de Éxito

Después de la implementación, monitorear:

Onboarding

- **Completación**: Objetivo >70% (antes: ~25%)
- **Time to First Value**: Objetivo <3 min (antes: ~12 min)
- **Activación D7**: Objetivo >65% (antes: ~35%)

Chatbot

- **Tasa de Resolución**: Objetivo >80%
- **Tiempo de Respuesta**: Objetivo <30 seg
- **Satisfacción (CSAT)**: Objetivo >4.5/5

Emails

- **Open Rate**: Objetivo >40%
- **Click Rate**: Objetivo >15%
- **Conversión**: Objetivo >10%

Mobile

- **Uso Móvil**: Objetivo >50% (antes: ~20%)
- **Bounce Rate Móvil**: Objetivo <30%
- **Session Duration**: Objetivo >10 min

🐛 Troubleshooting

Problema: Emails no se envían

Solución:
1. Verificar API key en `.env`
2. Revisar logs: `yarn prisma studio` > scheduled_emails
3. Verificar status de emails (pending/sent/failed)
4. Ejecutar manualmente: `curl .../api/cron/process-emails`

Problema: Chatbot no responde

Solución:
1. Verificar que `/api/chatbot` responde 200
2. Revisar console del navegador por errores
3. Verificar API key de GPT-4 (si aplica)
4. Fallback: Usar respuestas predefinidas

Problema: Bottom nav no aparece en móvil

Solución:
1. Verificar que está en layout root
2. Revisar breakpoint: debe ser `md:hidden`
3. Verificar z-index (debe ser >=100)
4. Inspeccionar con DevTools en modo móvil

📞 Soporte

- **Email**: soporte@inmova.app
- **Documentación**: Ver `ANALISIS_ZERO_TOUCH_ONBOARDING.md`
- **Web**: https://www.inmova.app

Fecha: Diciembre 2024
Versión: 1.0
Autor: Equipo INMOVA