Sistema de Notificaciones - Extensión del Schema

Resumen

Este documento describe la extensión del schema de Prisma para implementar un sistema completo de notificaciones en CitaPlanner.

Fecha de Implementación

Fecha: 2025-10-09
Versión: 1.4.0
Rama: feature/notifications-system

📋 Nuevos Enums

NotificationType (Extendido)

Tipos de notificaciones soportadas:
- `EMAIL` - Notificación por correo electrónico
- `SMS` - Notificación por SMS
- `WHATSAPP` - Notificación por WhatsApp
- `APPOINTMENT_REMINDER` - Recordatorio de cita
- `APPOINTMENT_CONFIRMATION` - Confirmación de cita
- `APPOINTMENT_CANCELLATION` - Cancelación de cita
- `APPOINTMENT_RESCHEDULE` - Cambio de horario de cita
- `PROMOTION` - Promoción u oferta
- `PAYMENT_REMINDER` - Recordatorio de pago

NotificationChannel (Nuevo)

Canales de entrega de notificaciones:
- `WHATSAPP` - WhatsApp (Evolution API o Cloud API)
- `PUSH` - Web Push Notifications
- `EMAIL` - Correo electrónico
- `SMS` - Mensaje de texto

NotificationStatus (Extendido)

Estados de las notificaciones:
- `PENDING` - Pendiente de envío
- `SENT` - Enviada
- `DELIVERED` - Entregada
- `FAILED` - Falló el envío
- `READ` - Leída por el destinatario

📊 Nuevos Modelos

1. NotificationSettings

Propósito: Configuración global de notificaciones por tenant.

Campos principales:
- `whatsappEnabled`, `pushEnabled`, `emailEnabled`, `smsEnabled` - Habilitar/deshabilitar canales
- `evolutionApiUrl`, `evolutionApiKey`, `whatsappInstanceName` - Configuración de Evolution API
- `appointmentReminderEnabled` - Habilitar recordatorios automáticos
- `appointmentReminderTimes` - JSON array con minutos de anticipación (ej: [1440, 60] = 24h y 1h antes)
- `autoConfirmEnabled` - Confirmación automática de citas

Relaciones:
- `tenant` (1:1) - Cada tenant tiene una configuración única

Casos de uso:
- Configurar canales de notificación activos
- Establecer credenciales de WhatsApp Evolution API
- Definir tiempos de recordatorios automáticos
- Activar/desactivar confirmaciones automáticas

---

2. NotificationTemplate

Propósito: Plantillas de mensajes personalizables para cada tipo de notificación.

Campos principales:
- `name` - Nombre descriptivo de la plantilla
- `type` - Tipo de notificación (NotificationType)
- `channel` - Canal de entrega (NotificationChannel)
- `subject` - Asunto (solo para emails)
- `message` - Plantilla del mensaje con variables
- `isActive` - Si la plantilla está activa
- `isDefault` - Si es la plantilla por defecto para este tipo+canal

Variables soportadas en plantillas:
- `{{clientName}}` - Nombre del cliente
- `{{appointmentDate}}` - Fecha de la cita
- `{{appointmentTime}}` - Hora de la cita
- `{{serviceName}}` - Nombre del servicio
- `{{professionalName}}` - Nombre del profesional
- `{{branchName}}` - Nombre de la sucursal
- `{{branchAddress}}` - Dirección de la sucursal
- `{{branchPhone}}` - Teléfono de la sucursal

Relaciones:
- `tenant` (N:1) - Múltiples plantillas por tenant

Casos de uso:
- Crear plantillas personalizadas por tipo de notificación
- Definir mensajes diferentes para cada canal (WhatsApp, Email, SMS, Push)

- Establecer plantillas por defecto
- Personalizar mensajes con variables dinámicas

Ejemplo de plantilla:

```
Hola {{clientName}},

Te recordamos tu cita para {{serviceName}} el {{appointmentDate}} a las {{appointment-
Time}} con {{professionalName}}.

Ubicación: {{branchName}}, {{branchAddress}}
Teléfono: {{branchPhone}}

¡Te esperamos!
```

---

3. NotificationLog

Propósito: Historial completo de todas las notificaciones enviadas.

Campos principales:
- `type` - Tipo de notificación
- `channel` - Canal utilizado
- `recipientId` - ID del cliente/usuario
- `recipientName` - Nombre completo del destinatario
- `recipientContact` - Teléfono/email/WhatsApp
- `subject` - Asunto (para emails)
- `message` - Mensaje enviado
- `status` - Estado actual
- `sentAt` , `deliveredAt` , `readAt` - Timestamps de seguimiento
- `errorMessage` - Mensaje de error si falló
- `metadata` - JSON con datos adicionales

Relaciones:
- `tenant` (N:1) - Múltiples logs por tenant
- `appointment` (N:1, opcional) - Cita relacionada
- `user` (N:1, opcional) - Usuario que envió (para notificaciones manuales)

Índices:
- `[tenantId, status]` - Búsqueda rápida por estado
- `[tenantId, createdAt]` - Ordenamiento cronológico
- `[appointmentId]` - Notificaciones de una cita específica

Casos de uso:
- Auditoría completa de notificaciones
- Seguimiento de entregas y lecturas
- Análisis de tasas de éxito/fallo
- Reenvío de notificaciones fallidas
- Reportes de actividad

---

4. PushSubscription

Propósito: Gestión de subscripciones de navegadores para Web Push Notifications.

Campos principales:
- `endpoint` - URL del servicio push (único)
- `p256dh` - Clave pública para encriptación
- `auth` - Secreto de autenticación
- `userAgent` - Información del navegador/dispositivo
- `isActive` - Si la subscripción está activa
- `lastUsedAt` - Última vez que se usó

Relaciones:
- `tenant` (N:1) - Múltiples subscripciones por tenant
- `user` (N:1, opcional) - Usuario asociado (si está autenticado)

Índices:
- `[tenantId, isActive]` - Subscripciones activas por tenant
- `[userId]` - Subscripciones de un usuario específico

Casos de uso:
- Registrar nuevas subscripciones de navegadores
- Enviar notificaciones push a dispositivos específicos
- Gestionar subscripciones inactivas o expiradas
- Soporte multi-dispositivo por usuario

Navegadores soportados:
- Chrome/Edge (Desktop y Android)
- Firefox (Desktop y Android)
- Safari (macOS y iOS 16.4+)
- Opera

🔄 Relaciones Actualizadas

Tenant

Nuevas relaciones:
- `notificationSettings` (1:1) - Configuración de notificaciones
- `pushSubscriptions` (1:N) - Subscripciones push del tenant

User

Nuevas relaciones:
- `pushSubscriptions` (1:N) - Subscripciones push del usuario

🚀 Flujo de Trabajo

1. Configuración Inicial

1. Admin configura `NotificationSettings` para el tenant
2. Habilita canales deseados (WhatsApp, Push, Email, SMS)

3. Configura credenciales de Evolution API si usa WhatsApp
4. Define tiempos de recordatorios automáticos

2. Gestión de Plantillas

1. Admin crea `NotificationTemplate` para cada tipo+canal
2. Personaliza mensajes con variables
3. Marca plantillas por defecto
4. Activa/desactiva plantillas según necesidad

3. Envío de Notificaciones

Automáticas (Recordatorios)

1. Sistema detecta citas próximas según `appointmentReminderTimes`
2. Busca plantilla por defecto para `APPOINTMENT_REMINDER` + canal activo
3. Reemplaza variables en la plantilla
4. Envía notificación por canal configurado
5. Registra en `NotificationLog`

Manuales

1. Usuario selecciona cliente/cita
2. Elige tipo de notificación y canal
3. Selecciona o personaliza plantilla
4. Sistema envía y registra en `NotificationLog`

4. Seguimiento

1. Sistema actualiza `status` en `NotificationLog`
2. Registra timestamps (`sentAt` , `deliveredAt` , `readAt`)
3. Captura errores en `errorMessage`
4. Admin puede ver historial y estadísticas

📱 Integración con Evolution API

Configuración

```
// En NotificationSettings
{
  whatsappEnabled: true,
  evolutionApiUrl: "https://api.evolution.com",
  evolutionApiKey: "your-api-key",
  whatsappInstanceName: "citaplanner-instance"
}
```

Endpoints Evolution API

- `POST /message/sendText` - Enviar mensaje de texto
- `POST /message/sendMedia` - Enviar imagen/archivo
- `GET /instance/connectionState` - Estado de conexión

🔔 Web Push Notifications

Registro de Subscripción

```
// Cliente (navegador)
const subscription = await registration.pushManager.subscribe({
  userVisibleOnly: true,
  applicationServerKey: vapidPublicKey
});

// Guardar en PushSubscription
await prisma.pushSubscription.create({
  data: {
    endpoint: subscription.endpoint,
    p256dh: subscription.keys.p256dh,
    auth: subscription.keys.auth,
    userAgent: navigator.userAgent,
    tenantId: currentTenant.id,
    userId: currentUser?.id
  }
});
```

Envío de Notificación Push

```
import webpush from 'web-push';

// Configurar VAPID keys
webpush.setVapidDetails(
  'mailto:admin@citaplanner.com',
  process.env.VAPID_PUBLIC_KEY,
  process.env.VAPID_PRIVATE_KEY
);

// Enviar a todas las subscripciones activas
const subscriptions = await prisma.pushSubscription.findMany({
  where: { tenantId, isActive: true }
});

for (const sub of subscriptions) {
  await webpush.sendNotification(
    {
      endpoint: sub.endpoint,
      keys: { p256dh: sub.p256dh, auth: sub.auth }
    },
    JSON.stringify({
      title: 'Recordatorio de Cita',
      body: message,
      icon: '/icon.png',
      badge: '/badge.png'
    })
  );
}
```

🔐 Seguridad y Privacidad

Datos Sensibles

- `evolutionApiKey` debe encriptarse en base de datos
- `p256dh` y `auth` de push subscriptions son seguros por diseño
- Logs de notificaciones contienen información personal (GDPR compliance)

Recomendaciones

1. Implementar encriptación para credenciales API
2. Establecer políticas de retención de logs
3. Permitir a usuarios opt-out de notificaciones
4. Cumplir con regulaciones de privacidad (GDPR, CCPA)

📊 Métricas y Reportes

KPIs Sugeridos

- Tasa de entrega por canal
- Tasa de lectura (para push y WhatsApp)
- Tiempo promedio de entrega
- Notificaciones fallidas por tipo de error
- Engagement por tipo de notificación

Queries Útiles

```
// Tasa de éxito por canal
const stats = await prisma.notificationLog.groupBy({
  by: ['channel', 'status'],
  where: { tenantId, createdAt: { gte: startDate } },
  _count: true
});

// Notificaciones de una cita
const logs = await prisma.notificationLog.findMany({
  where: { appointmentId },
  orderBy: { createdAt: 'desc' }
});
```

🛠️ Próximos Pasos

Fase 2: Implementación de APIs

1. Crear endpoints para CRUD de NotificationSettings
2. Implementar gestión de NotificationTemplate
3. Desarrollar servicio de envío de notificaciones
4. Integrar Evolution API para WhatsApp
5. Implementar Web Push con service workers

Fase 3: UI de Administración

1. Panel de configuración de notificaciones
2. Editor de plantillas con preview
3. Historial de notificaciones con filtros
4. Dashboard de métricas y estadísticas

Fase 4: Automatización

1. Cron jobs para recordatorios automáticos
2. Webhooks de Evolution API para estados
3. Reintento automático de notificaciones fallidas
4. Limpieza automática de logs antiguos

📚 Referencias

- Evolution API Documentation (https://doc.evolution-api.com/)
- Web Push Protocol (https://web.dev/push-notifications-overview/)
- Prisma Schema Reference (https://www.prisma.io/docs/reference/api-reference/prisma-schema-reference)
- WhatsApp Cloud API (https://developers.facebook.com/docs/whatsapp/cloud-api)

✅ Checklist de Migración

- [x] Extender enums existentes (NotificationType, NotificationStatus)
- [x] Crear nuevos enums (NotificationChannel)
- [x] Agregar modelo NotificationSettings
- [x] Actualizar modelo NotificationTemplate
- [x] Actualizar modelo NotificationLog
- [x] Agregar modelo PushSubscription
- [x] Actualizar relaciones en Tenant
- [x] Actualizar relaciones en User
- [x] Generar migración de Prisma
- [ ] Aplicar migración en desarrollo
- [ ] Aplicar migración en producción
- [ ] Crear seeds de datos de prueba
- [ ] Implementar APIs
- [ ] Desarrollar UI
- [ ] Configurar Evolution API
- [ ] Configurar VAPID keys para Push
- [ ] Testing completo

Autor: Sistema de Desarrollo CitaPlanner
Última actualización: 2025-10-09 07:30:00