Sistema de Automatización de Notificaciones

Descripción General

El sistema de automatización de notificaciones de CitaPlanner permite enviar notificaciones automáticas basadas en eventos del sistema de citas, sin intervención manual. Las notificaciones se envían a través de múltiples canales (WhatsApp, Push, Email, SMS) según la configuración del tenant.

Características Principales

1. Eventos que Disparan Notificaciones

Cita Creada → Confirmación

- Se envía automáticamente al crear una nueva cita
- Incluye detalles de la cita: fecha, hora, servicio, profesional, sucursal
- Canales: WhatsApp, Push, Email (según configuración)

Cita Próxima → Recordatorio

- Se envía automáticamente según tiempos configurados (ej: 24h y 1h antes)
- Verificación cada hora mediante cron job
- Prevención de duplicados (no envía si ya se envió recientemente)

Cita Modificada → Notificación de Cambio

- Se envía cuando cambia la fecha/hora de una cita
- Muestra fecha/hora anterior y nueva
- Solo se dispara si realmente cambió el horario

Cita Cancelada → Notificación de Cancelación

- Se envía cuando una cita es cancelada
- Informa al cliente sobre la cancelación

Pago Pendiente → Recordatorio de Pago

- Se envía para ventas con pagos pendientes
- Configurable para ejecutarse periódicamente

2. Sistema de Prevención de Duplicados

El sistema verifica el historial de notificaciones antes de enviar:

- **Confirmación**: No envía si ya se envió en las últimas 2 horas
- **Recordatorio**: No envía si ya se envió en las últimas 6 horas
- **Reprogramación**: No envía si ya se envió en la última hora
- **Cancelación**: No envía si ya se envió en la última hora
- **Pago**: No envía si ya se envió en las últimas 24 horas

3. Configuración de Recordatorios

Los recordatorios se configuran en `NotificationSettings`:

```
appointmentReminderTimes: string // JSON array: [1440, 60]
```

Valores en minutos antes de la cita:
- `1440` = 24 horas antes
- `60` = 1 hora antes
- `120` = 2 horas antes
- etc.

Ejemplo de configuración:

```
[1440, 120, 60]  // 24h, 2h y 1h antes
```

Arquitectura del Sistema

Componentes Principales

1. NotificationAutomationService

Ubicación: `app/lib/services/notificationAutomationService.ts`

Servicio principal que maneja toda la lógica de automatización:

```
class NotificationAutomationService {
  // Envía confirmación de cita
  async sendAppointmentConfirmation(appointmentId: string): Promise<void>

  // Verifica y envía recordatorios de citas próximas
  async sendAppointmentReminders(): Promise<{ sent, failed, skipped }>

  // Envía notificación de reprogramación
  async sendAppointmentRescheduleNotification(
    appointmentId: string,
    oldStartTime: Date,
    newStartTime: Date
  ): Promise<void>

  // Envía notificación de cancelación
  async sendAppointmentCancellationNotification(appointmentId: string): Promise<void>

  // Envía recordatorio de pago
  async sendPaymentReminder(saleId: string): Promise<void>
}
```

2. Notification Middleware

Ubicación: `app/lib/middleware/notificationMiddleware.ts`

Funciones helper para disparar notificaciones de forma asíncrona:

```
// Dispara confirmación sin bloquear respuesta
triggerAppointmentConfirmation(appointmentId: string): void

// Dispara reprogramación
triggerAppointmentReschedule(
  appointmentId: string,
  oldStartTime: Date,
  newStartTime: Date
): void

// Dispara cancelación
triggerAppointmentCancellation(appointmentId: string): void

// Dispara recordatorio de pago
triggerPaymentReminder(saleId: string): void
```

3. Notification Deduplication

Ubicación: `app/lib/utils/notificationDeduplication.ts`

Utilidades para prevenir duplicados:

```
// Verifica si existe notificación reciente
hasRecentNotification(
  type: NotificationType,
  recipientId: string,
  relatedId: string,
  hours: number
): Promise<boolean>

// Cuenta notificaciones en período
getNotificationCount(
  type: NotificationType,
  recipientId: string,
  hours: number
): Promise<number>
```

4. Cron Job Endpoint

Ubicación: `app/api/cron/send-reminders/route.ts`

Endpoint protegido para ejecutar recordatorios:

```
GET /api/cron/send-reminders
Authorization: Bearer <CRON_SECRET>

Response:
{
  success: true,
  message: "Recordatorios procesados exitosamente",
  stats: {
    sent: 15,
    failed: 0,
    skipped: 3
  },
  duration: "2345ms",
  timestamp: "2025-10-09T12:00:00Z"
}
```

5. Local Cron Scheduler (Desarrollo)

Ubicación: `app/lib/cron/scheduler.ts`

Sistema de cron jobs para desarrollo local usando `node-cron`:

```
class CronScheduler {
  start(): void  // Inicia todos los jobs
  stop(): void   // Detiene todos los jobs
  runJob(jobName: string): Promise<void>  // Ejecuta job manualmente
}
```

Jobs configurados:
- **Recordatorios**: Cada hora (`0 * * * *`)
- **Limpieza de logs**: Diario a las 3 AM (`0 3 * * *`)
- **Recordatorios de pago**: Cada 6 horas (`0 */6 * * *`)

Configuración

Variables de Entorno

Agregar a `.env`:

```
# Automatización de Notificaciones
NOTIFICATION_AUTOMATION_ENABLED=true
CRON_SECRET=tu_token_secreto_muy_seguro_aqui
NOTIFICATION_LOG_RETENTION_DAYS=90
```

Configuración en Vercel

1. Crear archivo `vercel.json` en la raíz del proyecto:

```
{
  "crons": [
    {
      "path": "/api/cron/send-reminders",
      "schedule": "0 * * * *"
    }
  ]
}
```

2. Configurar variables de entorno en Vercel:

1. Ve a tu proyecto en Vercel Dashboard
2. Settings → Environment Variables
3. Agrega:
   - `CRON_SECRET`: Token secreto para proteger el endpoint
   - `NOTIFICATION_AUTOMATION_ENABLED`: `true`
   - `NOTIFICATION_LOG_RETENTION_DAYS`: `90`

3. Desplegar:

```
vercel --prod
```

Configuración en Otros Servicios

Si no usas Vercel, puedes configurar un servicio externo para llamar al endpoint:

Usando cron-job.org:

1. Crea una cuenta en https://cron-job.org
2. Crea un nuevo cron job:
   - URL: `https://tu-dominio.com/api/cron/send-reminders`
   - Schedule: `0 * * * *` (cada hora)
   - Headers: `Authorization: Bearer tu_cron_secret`

Usando GitHub Actions:

```yaml
name: Send Appointment Reminders
on:
  schedule:
    - cron: '0 * * * *'  # Cada hora
  workflow_dispatch:  # Permite ejecución manual

jobs:
  send-reminders:
    runs-on: ubuntu-latest
    steps:
      - name: Call Cron Endpoint
        run: |
          curl -X GET \
            -H "Authorization: Bearer ${{ secrets.CRON_SECRET }}" \
            https://tu-dominio.com/api/cron/send-reminders
```

Uso en Desarrollo Local

1. Iniciar el Scheduler Local

El scheduler se inicia automáticamente en modo desarrollo si `NOTIFICATION_AUTOMATION_ENABLED=true`.

Para controlarlo manualmente:

```typescript
import { cronScheduler } from '@/lib/cron/scheduler';

// Iniciar
cronScheduler.start();

// Detener
cronScheduler.stop();

// Ejecutar job manualmente
await cronScheduler.runJob('reminders');
```

2. Probar Notificaciones

Crear una cita y verificar confirmación:

```
# Crear cita via API
curl -X POST http://localhost:3000/api/appointments \
  -H "Content-Type: application/json" \
  -d '{
    "clientId": "...",
    "serviceId": "...",
    "userId": "...",
    "branchId": "...",
    "startTime": "2025-10-10T10:00:00Z"
  }'

# Verificar logs
tail -f logs/notifications.log
```

Probar recordatorios manualmente:

```
# Llamar al endpoint de cron
curl -X GET http://localhost:3000/api/cron/send-reminders \
  -H "Authorization: Bearer tu_cron_secret"
```

3. Ejecutar Script de Limpieza

```
# Dry run (simulación)
npx ts-node app/scripts/cleanup-notification-logs.ts

# Ejecución real
npx ts-node app/scripts/cleanup-notification-logs.ts --execute

# Con parámetros personalizados
npx ts-node app/scripts/cleanup-notification-logs.ts --days=30 --execute
```

Integración con Endpoints Existentes

Appointments API

Los endpoints de citas ya están integrados:

POST /api/appointments

```
// Después de crear la cita
triggerAppointmentConfirmation(appointment.id);
```

PUT /api/appointments/[id]

```
// Si cambió la fecha/hora
if (updateData.startTime !== existingAppointment.startTime) {
  triggerAppointmentReschedule(
    appointmentId,
    existingAppointment.startTime,
    updateData.startTime
  );
}
```

DELETE /api/appointments/[id]

```
// Al cancelar
triggerAppointmentCancellation(appointmentId);
```

Monitoreo y Logs

Verificar Logs de Notificaciones

```
-- Últimas notificaciones enviadas
SELECT * FROM notification_logs
ORDER BY created_at DESC
LIMIT 50;

-- Estadísticas por tipo
SELECT type, status, COUNT(*) as count
FROM notification_logs
WHERE created_at > NOW() - INTERVAL '24 hours'
GROUP BY type, status;

-- Notificaciones fallidas
SELECT * FROM notification_logs
WHERE status = 'FAILED'
ORDER BY created_at DESC;
```

Logs de Aplicación

```
# Ver logs en tiempo real
tail -f logs/app.log | grep NotificationAutomation

# Buscar errores
grep "ERROR" logs/app.log | grep Notification
```

Troubleshooting

Las notificaciones no se envían

1. **Verificar configuración**:
   `sql`
   `SELECT * FROM notification_settings;`
   - Verificar que los canales estén habilitados
   - Verificar configuración de Evolution API (WhatsApp)

2. **Verificar variables de entorno**:
   ```bash
   echo $NOTIFICATION_AUTOMATION_ENABLED
   echo $CRON_SECRET
   ```

3. **Verificar logs**:
   ```bash
   grep "NotificationAutomation" logs/app.log
   ```

Los recordatorios no se ejecutan

1. **Verificar cron job**:
   - En Vercel: Settings → Cron Jobs
   - Verificar que el schedule sea correcto
   - Verificar últimas ejecuciones

2. **Probar manualmente**:
   ```bash
   curl -X GET https://tu-dominio.com/api/cron/send-reminders \
     -H "Authorization: Bearer $CRON_SECRET"
   ```

3. **Verificar tiempos de recordatorio**:
   ```sql
   SELECT appointment_reminder_times FROM notification_settings;
   ```

Notificaciones duplicadas

1. **Verificar sistema de deduplicación**:
   ```typescript
   // Debería retornar true si existe notificación reciente
   const isDuplicate = await hasRecentNotification(
     NotificationType.APPOINTMENT_REMINDER,
     clientId,
     appointmentId,
     6
   );
   ```

2. **Verificar logs duplicados**:
   ```sql
   SELECT recipient_id, type, COUNT(*) as count
   FROM notification_logs
   WHERE created_at > NOW() - INTERVAL '1 hour'
   GROUP BY recipient_id, type
   HAVING COUNT(*) > 1;
   ```

Error 401 en cron endpoint

- Verificar que el header `Authorization` sea correcto
- Verificar que `CRON_SECRET` esté configurado
- Formato: `Authorization: Bearer tu_cron_secret`

Mejores Prácticas

1. Configuración de Recordatorios

- **Recomendado**: `[1440, 60]` (24h y 1h antes)
- **Para servicios largos**: `[2880, 1440, 60]` (48h, 24h y 1h)
- **Para servicios cortos**: `[120, 30]` (2h y 30min)

2. Limpieza de Logs

- Ejecutar limpieza mensualmente
- Mantener logs de notificaciones fallidas para auditoría
- Retención recomendada: 90 días

3. Monitoreo

- Configurar alertas para notificaciones fallidas
- Revisar estadísticas semanalmente
- Monitorear tasa de entrega de WhatsApp

4. Testing

- Probar con citas de prueba antes de producción
- Verificar plantillas de notificaciones
- Probar todos los canales habilitados

Próximas Mejoras

- [ ] Sistema de colas con reintentos automáticos
- [ ] Priorización de notificaciones
- [ ] Notificaciones programadas personalizadas
- [ ] Dashboard de estadísticas de notificaciones
- [ ] A/B testing de plantillas
- [ ] Integración con más canales (Telegram, etc.)
- [ ] Notificaciones basadas en eventos de negocio
- [ ] Machine learning para optimizar tiempos de envío

Soporte

Para problemas o preguntas:
1. Revisar esta documentación
2. Verificar logs de aplicación
3. Consultar `NOTIFICATIONS_README.md` para arquitectura general
4. Revisar `NOTIFICATIONS_DEPLOYMENT.md` para deployment