

API de Notificaciones - CitaPlanner

Documentación completa de la API de notificaciones multicanal de CitaPlanner.

Tabla de Contenidos







1. [Visión General](#)
2. [Canales Soportados](#)
3. [Endpoints de Configuración](#)
4. [Endpoints de Plantillas](#)
5. [Endpoints de WhatsApp](#)
6. [Endpoints de Push Notifications](#)
7. [Endpoints de Logs](#)
8. [Modelos de Datos](#)
9. [Ejemplos de Uso](#)

Visión General

El sistema de notificaciones de CitaPlanner permite enviar mensajes a través de múltiples canales:


- **WhatsApp** (vía Evolution API)
- **Email** (SMTP)
- **SMS** (Twilio)
- **Push Notifications** (Web Push)

Características principales:


-  Gestión centralizada de notificaciones
-  Plantillas reutilizables con variables
-  Logs detallados de envíos
-  Webhooks para actualizaciones de estado
-  Envío masivo (bulk)
-  Configuración por tenant

Canales Soportados

WhatsApp

- **Estado:**  Implementado
- **Proveedor:** Evolution API
- **Características:** Texto, imágenes, documentos, webhooks

Email

- **Estado:**  Pendiente
- **Proveedor:** SMTP
- **Características:** HTML, adjuntos, plantillas

SMS

- **Estado:**  Pendiente

- **Proveedor:** Twilio
- **Características:** Texto simple, confirmación de entrega

Push Notifications

- **Estado:** ☒ Implementado
- **Proveedor:** Web Push API
- **Características:** Notificaciones en navegador, acciones, imágenes

Endpoints de Configuración

GET /api/notifications/settings

Obtiene la configuración actual de notificaciones.

Autenticación: Requerida

Response:

```
{
  "success": true,
  "data": {
    "id": "settings-id",
    "emailEnabled": true,
    "smsEnabled": false,
    "whatsappEnabled": true,
    "pushEnabled": true,
    "evolutionApiUrl": "https://api.evolution.com",
    "evolutionApiKey": "****",
    "whatsappInstanceName": "citaplanner",
    "smtpHost": "smtp.gmail.com",
    "smtpPort": 587,
    "smtpUser": "noreply@citaplanner.com",
    "twilioAccountSid": null,
    "twilioAuthToken": null,
    "twilioPhoneNumber": null
  }
}
```

PUT /api/notifications/settings

Actualiza la configuración de notificaciones.

Autenticación: Requerida

Body:

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

Response:

```
{
  "success": true,
  "data": {
    "id": "settings-id",
    "emailEnabled": true,
    "whatsappEnabled": true,
    "pushEnabled": true
  }
}
```



Endpoints de Plantillas

GET /api/notifications/templates

Lista todas las plantillas de notificación.

Autenticación: Requerida

Query Parameters:

- `type` (opcional): Filtrar por tipo (REMINDER, CONFIRMATION, CANCELLATION, etc.)
- `channel` (opcional): Filtrar por canal (WHATSAPP, EMAIL, SMS, PUSH)

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "template-id",
      "name": "Recordatorio de Cita",
      "type": "REMINDER",
      "channel": "WHATSAPP",
      "subject": "Recordatorio: Cita en {{businessName}}",
      "message": "Hola {{clientName}}, te recordamos tu cita el {{appointmentDate}} a las {{appointmentTime}}.",
      "variables": ["clientName", "businessName", "appointmentDate", "appointmentTime"],
      "isActive": true
    }
  ]
}
```

POST /api/notifications/templates

Crea una nueva plantilla.

Autenticación: Requerida

Body:

```
{
  "name": "Recordatorio de Cita",
  "type": "REMINDER",
  "channel": "WHATSAPP",
  "subject": "Recordatorio: Cita en {{businessName}}",
  "message": "Hola {{clientName}}, te recordamos tu cita el {{appointmentDate}} a las {{appointmentTime}}.",
  "variables": ["clientName", "businessName", "appointmentDate", "appointmentTime"],
  "isActive": true
}
```

PUT /api/notifications/templates/:id

Actualiza una plantilla existente.

DELETE /api/notifications/templates/:id

Elimina una plantilla.



Endpoints de WhatsApp

POST /api/notifications/whatsapp/send

Envía un mensaje de WhatsApp.

Autenticación: Requerida

Body:

```
{
  "to": "+5491234567890",
  "message": "Hola, este es un mensaje de prueba"
}
```

Response:

```
{
  "success": true,
  "data": {
    "messageId": "msg-123",
    "status": "sent"
  }
}
```

POST /api/notifications/whatsapp/webhook

Webhook para recibir actualizaciones de estado de mensajes de WhatsApp.

Autenticación: No requerida (validación por API key)

Body (ejemplo de Evolution API):

```
{
  "event": "messages.update",
  "instance": "citaplanner",
  "data": {
    "key": {
      "remoteJid": "5491234567890@s.whatsapp.net",
      "id": "msg-123"
    },
    "status": "READ"
  }
}
```



Endpoints de Push Notifications

POST /api/notifications/push/subscribe

Registra una nueva suscripción de push notification.

Autenticación: Requerida

Body:

```
{
  "endpoint": "https://fcm.googleapis.com/fcm/send/...",
  "keys": {
    "p256dh": "BNcRdreALRFXTk00UHK1EtK2wtaz5Ry4YfYCA_0QTpQtUbVlUls0VJXg7A8u-Ts1Xb-jhazAkj7I99e8QcYP7DkM=",
    "auth": "tBHItJI5svbpez7KI4CCXg=="
  },
  "userAgent": "Mozilla/5.0..."
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "subscription-id",
    "endpoint": "https://fcm.googleapis.com/fcm/send/..."
  }
}
```

DELETE /api/notifications/push/unsubscribe

Cancela una suscripción de push notification.

Autenticación: Requerida

Body:

```
{
  "endpoint": "https://fcm.googleapis.com/fcm/send/..."
}
```

Response:

```
{
  "success": true,
  "data": {
    "unsubscribed": true
  }
}
```

GET /api/notifications/push/subscriptions

Lista las subscripciones activas del usuario actual.

Autenticación: Requerida

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "subscription-id",
      "endpoint": "https://fcm.googleapis.com/fcm/send/...",
      "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/120.0.0.0",
      "lastUsedAt": "2024-01-15T10:30:00Z",
      "createdAt": "2024-01-10T08:00:00Z"
    }
  ]
}
```

POST /api/notifications/push/send

Envía una notificación push a un usuario específico.

Autenticación: Requerida

Body:

```
{
  "userId": "user-id",
  "title": "Nueva Cita Agendada",
  "body": "Tu cita ha sido confirmada para el 20 de enero a las 10:00 AM",
  "icon": "/icon-192x192.png",
  "url": "/appointments/123",
  "data": {
    "appointmentId": "123",
    "type": "appointment"
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "logId": "log-id",
    "total": 2,
    "sent": 2,
    "failed": 0
  }
}
```

POST /api/notifications/push/send-bulk

Envía notificaciones push a múltiples usuarios.

Autenticación: Requerida

Body:

```
{
  "userIds": ["user-1", "user-2", "user-3"],
  "title": "Mantenimiento Programado",
  "body": "El sistema estará en mantenimiento el 25 de enero de 2:00 AM a 4:00 AM",
  "icon": "/icon-maintenance.png",
  "url": "/announcements/maintenance"
}
```

Response:

```
{
  "success": true,
  "data": {
    "users": 3,
    "devices": 5,
    "sent": 5,
    "failed": 0
  }
}
```

Límites:

- Máximo 1000 usuarios por request
- Las subscripciones expiradas se marcan automáticamente como inactivas



Endpoints de Logs

GET /api/notifications/logs

Obtiene el historial de notificaciones enviadas.

Autenticación: Requerida

Query Parameters:

- `type` (opcional): Filtrar por tipo
- `channel` (opcional): Filtrar por canal
- `status` (opcional): Filtrar por estado (PENDING, SENT, DELIVERED, READ, FAILED)

- `recipientId` (opcional): Filtrar por destinatario
- `startDate` (opcional): Fecha inicio (ISO 8601)
- `endDate` (opcional): Fecha fin (ISO 8601)
- `page` (opcional): Número de página (default: 1)
- `limit` (opcional): Resultados por página (default: 50)

Response:

```
{
  "success": true,
  "data": {
    "logs": [
      {
        "id": "log-id",
        "type": "REMINDER",
        "channel": "PUSH",
        "recipientId": "user-id",
        "recipientName": "Juan Pérez",
        "recipientContact": "user-id",
        "subject": "Recordatorio de Cita",
        "message": "Tu cita es mañana a las 10:00 AM",
        "status": "SENT",
        "metadata": {
          "subscriptions": 2,
          "sent": 2,
          "failed": 0
        },
        "createdAt": "2024-01-15T10:00:00Z",
        "sentAt": "2024-01-15T10:00:01Z",
        "deliveredAt": null,
        "readAt": null
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 50,
      "total": 150,
      "pages": 3
    },
    "stats": {
      "PENDING": 5,
      "SENT": 120,
      "DELIVERED": 80,
      "READ": 45,
      "FAILED": 10
    }
  }
}
```

Modelos de Datos

NotificationSettings

```
{
  id: string;
  emailEnabled: boolean;
  smsEnabled: boolean;
  whatsappEnabled: boolean;
  pushEnabled: boolean;

  // WhatsApp (Evolution API)
  evolutionApiUrl?: string;
  evolutionApiKey?: string;
  whatsappInstanceName?: string;

  // Email (SMTP)
  smtpHost?: string;
  smtpPort?: number;
  smtpUser?: string;
  smtpPassword?: string;
  smtpFrom?: string;

  // SMS (Twilio)
  twilioAccountSid?: string;
  twilioAuthToken?: string;
  twilioPhoneNumber?: string;

  tenantId: string;
  createdAt: Date;
  updatedAt: Date;
}
```

NotificationTemplate

```
{
  id: string;
  name: string;
  type: NotificationType; // REMINDER, CONFIRMATION, CANCELLATION, etc.
  channel: NotificationChannel; // WHATSAPP, EMAIL, SMS, PUSH
  subject?: string;
  message: string;
  variables: string[]; // ["clientName", "appointmentDate", etc.]
  isActive: boolean;
  tenantId: string;
  createdAt: Date;
  updatedAt: Date;
}
```

NotificationLog

```
{
  id: string;
  type: NotificationType;
  channel: NotificationChannel;
  recipientId: string;
  recipientName: string;
  recipientContact: string;
  subject?: string;
  message: string;
  status: NotificationStatus; // PENDING, SENT, DELIVERED, READ, FAILED
  metadata?: Record<string, any>;
  tenantId: string;
  createdAt: Date;
  sentAt?: Date;
  deliveredAt?: Date;
  readAt?: Date;
}
```

PushSubscription

```
{
  id: string;
  endpoint: string; // Push service endpoint URL
  p256dh: string; // Public key for encryption
  auth: string; // Authentication secret
  userAgent?: string;
  isActive: boolean;
  lastUsedAt: Date;
  userId?: string;
  tenantId: string;
  createdAt: Date;
  updatedAt: Date;
}
```



Ejemplos de Uso

Ejemplo 1: Enviar Recordatorio de Cita por Push

```
// Cliente
const response = await fetch('/api/notifications/push/send', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    userId: 'user-123',
    title: 'Recordatorio de Cita',
    body: 'Tu cita es mañana a las 10:00 AM con Dr. García',
    icon: '/icon-appointment.png',
    url: '/appointments/456',
    data: {
      appointmentId: '456',
      type: 'reminder'
    }
  })
});

const result = await response.json();
console.log('Notificación enviada:', result);
```

Ejemplo 2: Suscribirse a Push Notifications

```
// Cliente - Usando el hook
import { usePushNotifications } from '@hooks/usePushNotifications';

function MyComponent() {
  const { subscribe, isSubscribed } = usePushNotifications();

  const handleSubscribe = async () => {
    const success = await subscribe();
    if (success) {
      alert('¡Subscrito exitosamente!');
    }
  };

  return (
    <button onClick={handleSubscribe} disabled={isSubscribed}>
      {isSubscribed ? 'Ya estás suscrito' : 'Activar notificaciones'}
    </button>
  );
}
```

Ejemplo 3: Enviar Notificación Masiva

```
// Servidor - Enviar a todos los usuarios con citas mañana
const usersWithAppointments = await prisma.appointment.findMany({
  where: {
    date: tomorrow,
    status: 'CONFIRMED'
  },
  select: {
    clientId: true
  }
});

const userIds = usersWithAppointments.map(a => a.clientId);

const response = await fetch('/api/notifications/push/send-bulk', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    userIds,
    title: 'Recordatorio de Cita',
    body: 'Tu cita es mañana. ¡No olvides asistir!',
    url: '/appointments'
  })
});
```

Ejemplo 4: Usar NotificationService Directamente

```
// Servidor
import { notificationService } from '@lib/services/notificationService';
import { NotificationType, NotificationChannel } from '@prisma/client';

// Enviar notificación push
const result = await notificationService.sendNotification({
  type: NotificationType.REMINDER,
  channel: NotificationChannel.PUSH,
  recipientId: 'user-123',
  message: 'Tu cita es mañana a las 10:00 AM',
  variables: {
    subject: 'Recordatorio de Cita',
    url: '/appointments/456',
    data: {
      appointmentId: '456'
    }
  }
});

console.log('Resultado:', result);
```



Autenticación

Todos los endpoints (excepto webhooks) requieren autenticación mediante NextAuth session.

Headers requeridos:

Cookie: next-auth.session-token=...

Validación:

```
const session = await getSession(authOptions);
if (!session || !session.user) {
  return NextResponse.json({ error: 'Unauthorized' }, { status: 401 });
}
```

Manejo de Errores

Todos los endpoints retornan respuestas en formato estándar:

Éxito:

```
{
  "success": true,
  "data": { ... }
}
```

Error:

```
{
  "success": false,
  "error": "Mensaje de error descriptivo"
}
```

Códigos de estado HTTP:

- 200 : Éxito
- 400 : Bad Request (datos inválidos)
- 401 : Unauthorized (no autenticado)
- 404 : Not Found (recurso no encontrado)
- 500 : Internal Server Error



Referencias

- [Evolution API Documentation](https://doc.evolution-api.com/) (https://doc.evolution-api.com/)
- [Web Push Protocol](https://datatracker.ietf.org/doc/html/rfc8030) (https://datatracker.ietf.org/doc/html/rfc8030)
- [Service Worker API](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API) (https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API)
- [Push API](https://developer.mozilla.org/en-US/docs/Web/API/Push_API) (https://developer.mozilla.org/en-US/docs/Web/API/Push_API)