

Sprint 1 - Fase 3: Endpoint API /api/services con CRUD completo

Versión: v1.8.6

Fecha: 15 de octubre de 2025













PR: #111

Estado:  Implementado y listo para producción

Resumen Ejecutivo

Se ha implementado un endpoint API completo para la gestión de servicios (`/api/services`) con todas las operaciones CRUD (Crear, Leer, Actualizar, Eliminar). Este endpoint proporciona una interfaz robusta y segura para gestionar los servicios ofrecidos en CitaPlanner, con validaciones de negocio completas, autenticación, multi-tenancy y control de permisos.

Objetivos Cumplidos

-  Endpoint GET para listar servicios con filtros avanzados
 -  Endpoint POST para crear nuevos servicios
 -  Endpoint GET por ID para obtener un servicio específico
 -  Endpoint PUT para actualizar servicios existentes
 -  Endpoint DELETE para desactivar servicios (soft delete)
 -  Autenticación y autorización con NextAuth
 -  Validaciones de negocio completas
 -  Multi-tenancy (filtrado por tenant)
 -  Control de permisos por rol
 -  Manejo de errores robusto
 -  Respuestas JSON estandarizadas
 -  Paginación opcional
-

Documentación de Endpoints

1. GET /api/services

Descripción: Obtiene todos los servicios del tenant actual.

Autenticación:  Requerida

Query Parameters (opcionales):

- `isActive` (boolean): Filtrar por estado activo/inactivo
- `category` (string): Filtrar por ID de categoría
- `search` (string): Buscar por nombre o descripción

- `limit` (number): Límite de resultados para paginación
- `offset` (number): Offset para paginación

Ejemplo de Request:

```
GET /api/services?isActive=true&search=corte&limit=10&offset=0
Authorization: Bearer {token}
```

Ejemplo de Response Exitosa (200):

```
{
  "success": true,
  "data": [
    {
      "id": "clxyz123",
      "name": "Corte de Cabello",
      "description": "Corte de cabello para caballeros",
      "duration": 30,
      "price": 150.00,
      "isActive": true,
      "color": "#3B82F6",
      "categoryId": "cat123",
      "tenantId": "tenant123",
      "createdAt": "2025-10-15T10:00:00.000Z",
      "updatedAt": "2025-10-15T10:00:00.000Z",
      "category": {
        "id": "cat123",
        "name": "Peluquería",
        "color": "#3B82F6"
      },
      "serviceUsers": [
        {
          "id": "su123",
          "serviceId": "clxyz123",
          "userId": "user123",
          "commission": 15.5,
          "user": {
            "id": "user123",
            "firstName": "Juan",
            "lastName": "Pérez",
            "email": "juan@example.com"
          }
        }
      ]
    }
  ],
  "meta": {
    "total": 25,
    "limit": 10,
    "offset": 0
  }
}
```

Códigos de Error:

- 401 : No autorizado (sin sesión o sin tenantId)
 - 500 : Error interno del servidor
-

2. POST /api/services

Descripción: Crea un nuevo servicio.

Autenticación:  Requerida

Body Parameters:

Requeridos:

- `name` (string): Nombre del servicio
- `duration` (number): Duración en minutos (entre 5 y 480)
- `price` (number): Precio del servicio (entre 0 y 999999.99)

Opcionales:

- `description` (string): Descripción del servicio
- `categoryId` (string): ID de la categoría
- `color` (string): Color hexadecimal (default: #3B82F6)

Ejemplo de Request:

```
POST /api/services
Authorization: Bearer {token}
Content-Type: application/json

{
  "name": "Manicure Completo",
  "description": "Manicure con esmaltado permanente",
  "duration": 60,
  "price": 250.00,
  "categoryId": "cat456",
  "color": "#EC4899"
}
```

Ejemplo de Response Exitosa (201):

```
{
  "success": true,
  "data": {
    "id": "clxyz456",
    "name": "Manicure Completo",
    "description": "Manicure con esmaltado permanente",
    "duration": 60,
    "price": 250.00,
    "isActive": true,
    "color": "#EC4899",
    "categoryId": "cat456",
    "tenantId": "tenant123",
    "createdAt": "2025-10-15T11:00:00.000Z",
    "updatedAt": "2025-10-15T11:00:00.000Z",
    "category": {
      "id": "cat456",
      "name": "Belleza",
      "color": "#EC4899"
    }
  }
}
```

Códigos de Error:

- 400 : Datos inválidos o servicio duplicado
- 401 : No autorizado
- 500 : Error interno del servidor

Validaciones:

- Nombre requerido y único por tenant
 - Duración entre 5 y 480 minutos (8 horas)
 - Precio entre 0 y 999999.99
 - CategoryId debe existir y pertenecer al tenant (si se proporciona)
-

3. GET /api/services/[id]

Descripción: Obtiene un servicio específico por ID.

Autenticación: ☒ Requerida

Parámetros de Ruta:

- id (string): ID del servicio

Ejemplo de Request:

```
GET /api/services/clxyz123
Authorization: Bearer {token}
```

Ejemplo de Response Exitosa (200):

```

{
  "success": true,
  "data": {
    "id": "clxyz123",
    "name": "Corte de Cabello",
    "description": "Corte de cabello para caballeros",
    "duration": 30,
    "price": 150.00,
    "isActive": true,
    "color": "#3B82F6",
    "categoryId": "cat123",
    "tenantId": "tenant123",
    "createdAt": "2025-10-15T10:00:00.000Z",
    "updatedAt": "2025-10-15T10:00:00.000Z",
    "category": {
      "id": "cat123",
      "name": "Peluquería",
      "color": "#3B82F6"
    },
  },
  "serviceUsers": [
    {
      "id": "su123",
      "serviceId": "clxyz123",
      "userId": "user123",
      "commission": 15.5,
      "user": {
        "id": "user123",
        "firstName": "Juan",
        "lastName": "Pérez",
        "email": "juan@example.com",
        "role": "PROFESSIONAL"
      }
    }
  ],
  "appointments": [
    {
      "id": "apt123",
      "startTime": "2025-10-16T14:00:00.000Z",
      "endTime": "2025-10-16T14:30:00.000Z",
      "status": "SCHEDULED"
    }
  ]
}

```

Códigos de Error:

- 401 : No autorizado
- 404 : Servicio no encontrado o no pertenece al tenant
- 500 : Error interno del servidor

4. PUT /api/services/[id]

Descripción: Actualiza un servicio existente.

Autenticación: ☒ Requerida

Permisos: ADMIN, SUPER_ADMIN, MANAGER

Parámetros de Ruta:

- `id` (string): ID del servicio

Body Parameters (todos opcionales):

- `name` (string): Nombre del servicio
- `description` (string): Descripción del servicio
- `duration` (number): Duración en minutos (entre 5 y 480)
- `price` (number): Precio del servicio (entre 0 y 999999.99)
- `categoryId` (string): ID de la categoría (null para remover)
- `color` (string): Color hexadecimal
- `isActive` (boolean): Estado activo/inactivo

Ejemplo de Request:

```
PUT /api/services/clxyz123
Authorization: Bearer {token}
Content-Type: application/json

{
  "name": "Corte de Cabello Premium",
  "price": 200.00,
  "description": "Corte de cabello premium con lavado incluido"
}
```

Ejemplo de Response Exitosa (200):

```
{
  "success": true,
  "data": {
    "id": "clxyz123",
    "name": "Corte de Cabello Premium",
    "description": "Corte de cabello premium con lavado incluido",
    "duration": 30,
    "price": 200.00,
    "isActive": true,
    "color": "#3B82F6",
    "categoryId": "cat123",
    "tenantId": "tenant123",
    "createdAt": "2025-10-15T10:00:00.000Z",
    "updatedAt": "2025-10-15T12:00:00.000Z",
    "category": {
      "id": "cat123",
      "name": "Peluquería",
      "color": "#3B82F6"
    },
    "serviceUsers": [...]
  }
}
```

Códigos de Error:

- 400 : Datos inválidos o servicio duplicado
- 401 : No autorizado
- 403 : Sin permisos (rol no autorizado)
- 404 : Servicio no encontrado o no pertenece al tenant
- 500 : Error interno del servidor

Validaciones:

- Nombre único por tenant (si se cambia)
- Duración entre 5 y 480 minutos (si se proporciona)
- Precio entre 0 y 999999.99 (si se proporciona)
- CategoryId debe existir y pertenecer al tenant (si se proporciona)

5. DELETE /api/services/[id]

Descripción: Elimina (desactiva) un servicio mediante soft delete.

Autenticación: ☒ Requerida

Permisos: ADMIN, SUPER_ADMIN, MANAGER

Parámetros de Ruta:

- `id` (string): ID del servicio

Ejemplo de Request:

```
DELETE /api/services/clxyz123
Authorization: Bearer {token}
```

Ejemplo de Response Exitosa (200):

```
{
  "success": true,
  "data": {
    "id": "clxyz123",
    "name": "Corte de Cabello",
    "description": "Corte de cabello para caballeros",
    "duration": 30,
    "price": 150.00,
    "isActive": false,
    "color": "#3B82F6",
    "categoryId": "cat123",
    "tenantId": "tenant123",
    "createdAt": "2025-10-15T10:00:00.000Z",
    "updatedAt": "2025-10-15T13:00:00.000Z",
    "category": {
      "id": "cat123",
      "name": "Peluquería",
      "color": "#3B82F6"
    }
  },
  "message": "Servicio desactivado exitosamente"
}
```

Códigos de Error:




- 400 : No se puede desactivar (tiene citas futuras)
- 401 : No autorizado
- 403 : Sin permisos (rol no autorizado)
- 404 : Servicio no encontrado o no pertenece al tenant
- 500 : Error interno del servidor

Validaciones:




- Verifica que no haya citas futuras (SCHEDULED o CONFIRMED) asociadas al servicio
 - Si hay citas futuras, retorna error 400 con el número de citas
-

Seguridad Implementada




Autenticación

-  Todos los endpoints requieren autenticación con NextAuth
-  Verificación de sesión válida en cada request
-  Validación de tenantId presente en la sesión






Multi-Tenancy

-  Filtrado automático por tenantId en todas las operaciones
-  Prevención de acceso a datos de otros tenants
-  Validación de pertenencia en operaciones de modificación

Control de Permisos

-  PUT y DELETE requieren roles: ADMIN, SUPER_ADMIN, MANAGER
-  Validación de roles a nivel de endpoint
-  Respuesta 403 Forbidden para usuarios sin permisos

Validación de Datos

-  Validación de tipos de datos
 -  Validación de rangos (duración, precio)
 -  Validación de unicidad (nombre por tenant)
 -  Validación de integridad referencial (categoryId)
 -  Prevención de soft delete con citas futuras
-

✓ Validaciones de Negocio

Creación de Servicio (POST)

Campo	Validación	Mensaje de Error
name	Requerido, string, no vacío	"Nombre, duración y precio son requeridos"
name	Único por tenant (case-insensitive)	"Ya existe un servicio con este nombre"
duration	Requerido, número entre 5 y 480	"La duración debe ser un número entre 5 y 480 minutos (8 horas)"
price	Requerido, número entre 0 y 999999.99	"El precio debe ser un número entre 0 y 999999.99"
categoryId	Debe existir y pertenecer al tenant	"La categoría especificada no existe o no pertenece a tu cuenta"

Actualización de Servicio (PUT)

Campo	Validación	Mensaje de Error
id	Debe existir y pertenecer al tenant	"Servicio no encontrado"
role	ADMIN, SUPER_ADMIN o MANAGER	"No tienes permisos para actualizar servicios"
name	Único por tenant si se cambia	"Ya existe otro servicio con este nombre"
duration	Número entre 5 y 480 (si se proporciona)	"La duración debe ser un número entre 5 y 480 minutos (8 horas)"
price	Número entre 0 y 999999.99 (si se proporciona)	"El precio debe ser un número entre 0 y 999999.99"
categoryId	Debe existir y pertenecer al tenant	"La categoría especificada no existe o no pertenece a tu cuenta"

Eliminación de Servicio (DELETE)

Validación	Mensaje de Error
Servicio existe y pertenece al tenant	"Servicio no encontrado"
Rol autorizado	"No tienes permisos para eliminar servicios"
Sin citas futuras SCHEDULED/CONFIRMED	"No se puede desactivar el servicio porque tiene X cita(s) futura(s) programada(s)"



Estructura de Datos

Modelo Service (Prisma)

```

model Service {
  id          String  @id @default(cuid())
  name        String
  description String?
  duration    Int // en minutos
  price       Float
  isActive    Boolean @default(true)
  color       String  @default("#3B82F6")

  tenantId String
  tenant   Tenant @relation(fields: [tenantId], references: [id], onDelete: Cascade)

  categoryId String?
  category   ServiceCategory? @relation(fields: [categoryId], references: [id], onDelete: SetNull)

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  // Relations
  appointments Appointment[]
  serviceUsers ServiceUser[]
}

```



Casos de Uso

1. Listar servicios activos

```

const response = await fetch('/api/services?isActive=true', {
  method: 'GET',
  headers: {
    'Authorization': `Bearer ${token}`
  }
});
const { data, meta } = await response.json();

```

2. Buscar servicios por nombre

```
const response = await fetch('/api/services?search=corte', {
  method: 'GET',
  headers: {
    'Authorization': `Bearer ${token}`
  }
});
const { data } = await response.json();
```

3. Crear nuevo servicio

```
const response = await fetch('/api/services', {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    name: 'Manicure Completo',
    duration: 60,
    price: 250.00,
    description: 'Manicure con esmaltado permanente',
    categoryId: 'cat123'
  })
});
const { data } = await response.json();
```

4. Actualizar precio de servicio

```
const response = await fetch('/api/services/clxyz123', {
  method: 'PUT',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    price: 200.00
  })
});
const { data } = await response.json();
```

5. Desactivar servicio

```
const response = await fetch('/api/services/clxyz123', {
  method: 'DELETE',
  headers: {
    'Authorization': `Bearer ${token}`
  }
});
const { data, message } = await response.json();
```

Testing Manual

Preparación

1. Obtener token de autenticación válido
2. Identificar tenantId de la sesión
3. Crear categoría de servicio (opcional)

Pruebas Recomendadas

GET /api/services

- [] Listar todos los servicios
- [] Filtrar por isActive=true
- [] Filtrar por isActive=false
- [] Buscar por nombre
- [] Filtrar por categoría
- [] Paginación con limit y offset
- [] Sin autenticación (debe retornar 401)

POST /api/services

- [] Crear servicio válido
- [] Crear sin nombre (debe retornar 400)
- [] Crear sin duración (debe retornar 400)
- [] Crear sin precio (debe retornar 400)
- [] Crear con duración < 5 (debe retornar 400)
- [] Crear con duración > 480 (debe retornar 400)
- [] Crear con precio negativo (debe retornar 400)
- [] Crear con nombre duplicado (debe retornar 400)
- [] Crear con categoryId inválido (debe retornar 400)
- [] Sin autenticación (debe retornar 401)

GET /api/services/[id]

- [] Obtener servicio existente
- [] Obtener servicio inexistente (debe retornar 404)
- [] Obtener servicio de otro tenant (debe retornar 404)
- [] Sin autenticación (debe retornar 401)

PUT /api/services/[id]

- [] Actualizar nombre
- [] Actualizar precio
- [] Actualizar duración
- [] Actualizar descripción
- [] Actualizar categoryId
- [] Remover categoryId (null)
- [] Actualizar isActive
- [] Actualizar con nombre duplicado (debe retornar 400)
- [] Actualizar con duración inválida (debe retornar 400)
- [] Actualizar con precio inválido (debe retornar 400)

- [] Sin autenticación (debe retornar 401)
- [] Con rol no autorizado (debe retornar 403)
- [] Servicio inexistente (debe retornar 404)

DELETE /api/services/[id]

- [] Desactivar servicio sin citas futuras
- [] Intentar desactivar con citas futuras (debe retornar 400)
- [] Sin autenticación (debe retornar 401)
- [] Con rol no autorizado (debe retornar 403)
- [] Servicio inexistente (debe retornar 404)



Notas Técnicas

Soft Delete

- Los servicios no se eliminan físicamente de la base de datos
- Se marca el campo `isActive` como `false`
- Esto preserva la integridad referencial con citas históricas
- Los servicios inactivos no aparecen en listados por defecto

Paginación

- Implementada mediante `limit` y `offset`
- Retorna metadata con total de registros
- Útil para cargar servicios en lotes en el frontend

Inclusión de Relaciones

- GET /api/services incluye: category, serviceUsers (con user)
- GET /api/services/[id] incluye: category, serviceUsers, appointments (próximos 5)
- PUT incluye: category, serviceUsers en la respuesta

Búsqueda Case-Insensitive

- La búsqueda por nombre es insensible a mayúsculas/minúsculas
- Utiliza `mode: 'insensitive'` en Prisma
- Funciona tanto en name como en description






Integración con el Sistema

Módulos Afectados

- **✓ CRM:** Los servicios pueden asignarse a citas de clientes
- **✓ Calendario:** Los servicios determinan la duración de las citas
- **✓ POS/Ventas:** Los servicios pueden venderse como productos
- **✓ Profesionales:** Los servicios se asignan a profesionales mediante serviceUsers
- **✓ Reportes:** Los servicios se incluyen en reportes de ingresos y citas

Dependencias

-  **Prisma Client:** Para acceso a base de datos
-  **NextAuth:** Para autenticación y sesión
-  **Next.js Route Handlers:** Para manejo de requests HTTP



Mejoras Futuras (Fuera de Scope)

Las siguientes mejoras podrían implementarse en fases futuras:

1. **Búsqueda Avanzada**
 - Búsqueda full-text con relevancia
 - Filtros combinados múltiples
 - Ordenamiento personalizado
2. **Versionado de Servicios**
 - Histórico de cambios de precios
 - Auditoría de modificaciones
 - Restauración de versiones anteriores
3. **Servicios Compuestos**
 - Paquetes de múltiples servicios
 - Descuentos por combo
 - Duraciones combinadas
4. **Disponibilidad por Sucursal**
 - Servicios específicos por sucursal
 - Precios diferentes por ubicación
 - Horarios de disponibilidad
5. **Imágenes de Servicios**
 - Galería de fotos
 - Imagen destacada
 - Storage en cloud



Debugging

Logs

Todos los endpoints incluyen logging de errores:

```
console.error('Error fetching services:', error);
console.error('Error creating service:', error);
console.error('Error updating service:', error);
console.error('Error deleting service:', error);
```

Common Issues

Error 401: No autorizado

- Verificar que la sesión esté activa

- Verificar que el token sea válido
- Verificar que tenantId esté presente en session.user

Error 403: Forbidden

- Verificar que el usuario tenga rol ADMIN, SUPER_ADMIN o MANAGER
- Solo aplica para PUT y DELETE

Error 404: Service not found

- Verificar que el ID sea correcto
- Verificar que el servicio pertenezca al tenant actual
- Verificar que el servicio exista en la base de datos

Error 400: Validation error

- Revisar mensaje de error específico
- Verificar que todos los campos requeridos estén presentes
- Verificar que los valores estén en los rangos permitidos




Checklist de Implementación

- [x] Crear app/api/services/route.ts (GET, POST)
- [x] Crear app/api/services/[id]/route.ts (GET, PUT, DELETE)
- [x] Implementar autenticación con NextAuth
- [x] Implementar filtrado multi-tenant
- [x] Implementar validaciones de negocio
- [x] Implementar control de permisos por rol
- [x] Implementar soft delete
- [x] Implementar paginación
- [x] Implementar búsqueda y filtros
- [x] Incluir relaciones (category, serviceUsers)
- [x] Manejo de errores robusto
- [x] Respuestas JSON estandarizadas
- [x] Logging de errores
- [x] Documentación completa
- [x] Commit y push a rama feature



Conclusión

La implementación del endpoint `/api/services` proporciona una base sólida para la gestión de servicios en CitaPlanner. Con validaciones robustas, seguridad multi-tenant, control de permisos y una API RESTful completa, este endpoint está listo para integrarse con el frontend y soportar todas las operaciones de gestión de servicios del sistema.

Estado:  Listo para merge a main

Próximo paso: Crear PR #111 y mergear a producción

Autor: DeepAgent

Sprint: Sprint 1 - Fase 3

Versión CitaPlanner: v1.8.6