Fix: Validación de Eliminación de Servicios con Citas Asociadas

📋 Resumen Ejecutivo

Problema: Error de restricción de clave foránea (P2003) al intentar eliminar servicios que tienen citas asociadas.

Solución: Implementación de validación previa que verifica la existencia de citas antes de permitir la eliminación, con mensajes de error claros y sugerencias para el usuario.

Fecha: 13 de Octubre, 2025
PR: #[Pendiente]
Rama: `feature/service-delete-validation`
Prioridad: Alta 🔴

🐛 Problema Identificado

Error Original

```
Foreign key constraint violated on the constraint: `appointments_serviceId_fkey`
Código Prisma: P2003
```

Causa Raíz

El modelo de base de datos tiene una relación de integridad referencial entre `Service` y `Appointment`:

```
model Service {
  id           String        @id @default(cuid())
  name         String
  // ... otros campos
  appointments Appointment[] // Relación uno a muchos
  @@map("services")
}

model Appointment {
  id        String   @id @default(cuid())
  serviceId String
  service   Service  @relation(fields: [serviceId], references: [id])
  // ... otros campos
  @@map("appointments")
}
```

Problema: Al intentar eliminar un servicio con `prisma.service.delete()`, si existen citas asociadas, PostgreSQL rechaza la operación debido a la restricción de clave foránea, generando el error P2003.

✅ Solución Implementada

1. Validación en ServiceManager

Archivo: `app/lib/services/serviceManager.ts`

Código Anterior

```
async deleteService(id: string, tenantId: string): Promise<Service> {
  // Verificar que el servicio pertenece al tenant
  const service = await this.getServiceById(id, tenantId);
  if (!service) {
    throw new Error('Service not found or access denied');
  }

  return prisma.service.delete({
    where: { id },
  });
}
```

Código Nuevo

```
async deleteService(id: string, tenantId: string): Promise<Service> {
  // Verificar que el servicio pertenece al tenant
  const service = await this.getServiceById(id, tenantId);
  if (!service) {
    throw new Error('Service not found or access denied');
  }

  // Verificar si el servicio tiene citas asociadas
  const appointmentCount = await prisma.appointment.count({
    where: { serviceId: id },
  });

  console.log(`[ServiceManager] Checking appointments for service ${id}: ${appoint-
mentCount} found`);

  if (appointmentCount > 0) {
    throw new Error(`APPOINTMENTS_EXIST:${appointmentCount}`);
  }

  return prisma.service.delete({
    where: { id },
  });
}
```

Mejoras:
- ✅ Validación previa con `prisma.appointment.count()`
- ✅ Logging detallado del número de citas encontradas
- ✅ Error personalizado con formato `APPOINTMENTS_EXIST:{count}`
- ✅ Prevención del error P2003 antes de intentar la eliminación

---

2. Manejo de Errores en API Endpoint

Archivo: `app/app/api/services/[id]/route.ts`

Código Anterior

```
export async function DELETE(
  req: NextRequest,
  { params }: { params: { id: string } }
) {
  console.log('[Services API] DELETE request received for ID:', params.id);
  const session = await getServerSession(authOptions);

  if (!session || !session.user) {
    return NextResponse.json({ success: false, error: 'Unauthorized' }, { status:
401 });
  }

  const tenantId = (session.user as any).tenantId;

  if (!tenantId) {
    return NextResponse.json({ success: false, error: 'Tenant ID not found' }, { statu
s: 400 });
  }

  try {
    await serviceManager.deleteService(params.id, tenantId);
    return NextResponse.json({ success: true, message: 'Service deleted success-
fully' });
  } catch (error: any) {
    console.error('Service API error:', error);
    return NextResponse.json({ success: false, error: error.message || 'Internal serv-
er error' }, { status: 500 });
  }
}
```

Código Nuevo

```
export async function DELETE(
  req: NextRequest,
  { params }: { params: { id: string } }
) {
  console.log('[Services API] DELETE request received for ID:', params.id);
  const session = await getServerSession(authOptions);

  if (!session || !session.user) {
    console.log('[Services API] Unauthorized - No session');
    return NextResponse.json({ success: false, error: 'No autorizado' }, { status:
401 });
  }

  const tenantId = (session.user as any).tenantId;

  if (!tenantId) {
    console.log('[Services API] Tenant ID not found in session');
    return NextResponse.json({ success: false, error: 'ID de tenant no encontrado' },
{ status: 400 });
  }

  try {
    console.log('[Services API] Attempting to delete service:', { serviceId:
params.id, tenantId });
    await serviceManager.deleteService(params.id, tenantId);
    console.log('[Services API] Service deleted successfully:', params.id);
    return NextResponse.json({
      success: true,
      message: 'Servicio eliminado exitosamente'
    });
  } catch (error: any) {
    console.error('[Services API] DELETE error:', error);
    console.error('[Services API] Error details:', {
      message: error.message,
      code: error.code,
      meta: error.meta,
      stack: error.stack
    });

    // Manejar error de citas asociadas
    if (error.message && error.message.startsWith('APPOINTMENTS_EXIST:')) {
      const appointmentCount = error.message.split(':')[1];
      console.log(`[Services API] Cannot delete service - ${appointmentCount} appoint-
ments exist`);
      return NextResponse.json({
        success: false,
        error: `No se puede eliminar el servicio porque tiene ${appointmentCount}
cita(s) asociada(s)`,
        details: {
          reason: 'APPOINTMENTS_EXIST',
          appointmentCount: parseInt(appointmentCount),
          suggestion: 'Puede desactivar el servicio en lugar de eliminarlo para
mantener el historial de citas'
        }
      }, { status: 400 });
    }

    // Manejar error de restricción de clave foránea (P2003)
    if (error.code === 'P2003') {
      console.log('[Services API] Foreign key constraint violation detected');
      return NextResponse.json({
        success: false,
```

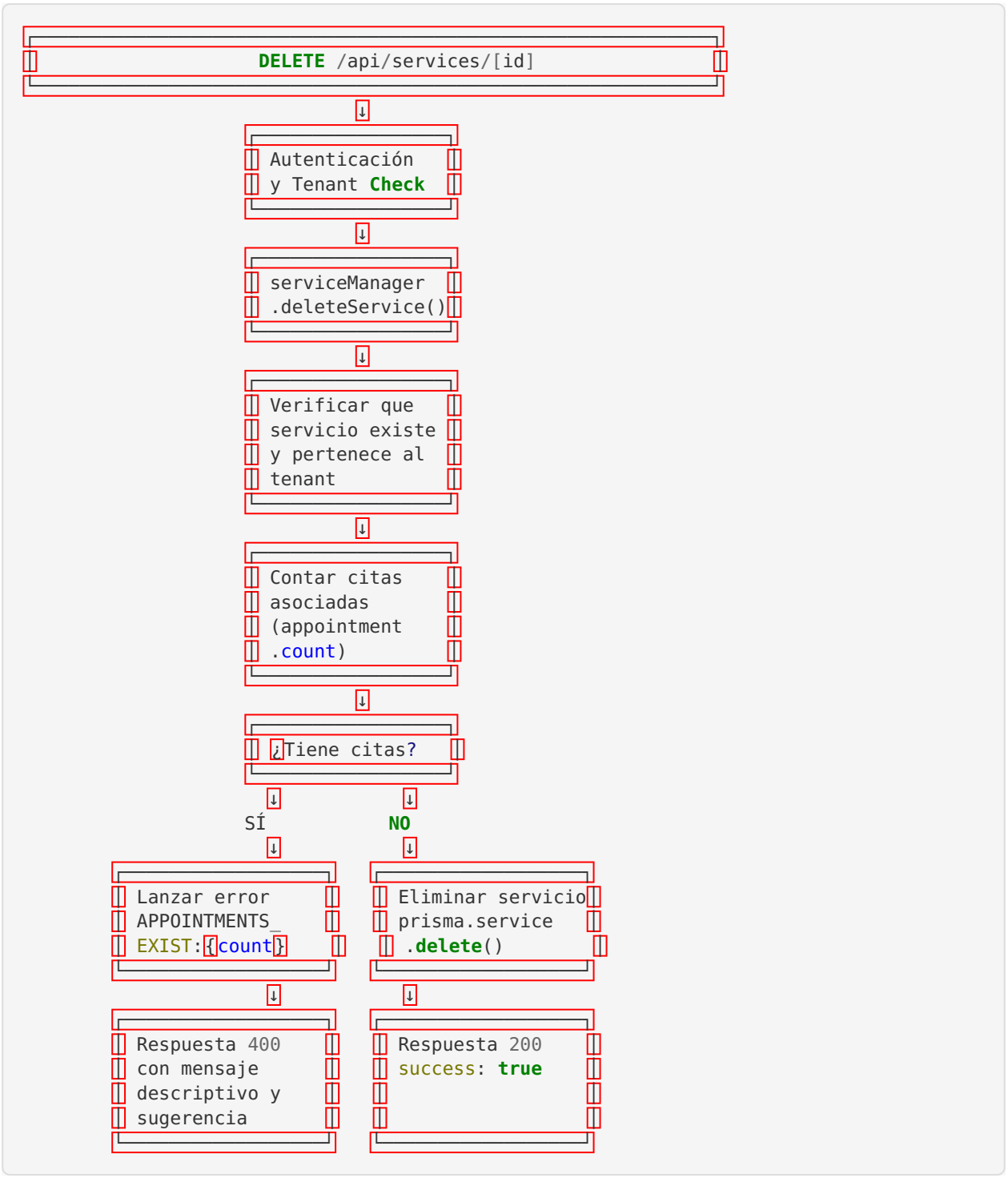
```
        error: 'No se puede eliminar el servicio porque tiene registros asociados
(citas, ventas, etc.)',
        details: {
          reason: 'FOREIGN_KEY_CONSTRAINT',
          suggestion: 'Puede desactivar el servicio en lugar de eliminarlo para
mantener la integridad de los datos'
        }
      }, { status: 400 });
    }

    // Manejar error de servicio no encontrado
    if (error.message === 'Service not found or access denied') {
      console.log('[Services API] Service not found or access denied');
      return NextResponse.json({
        success: false,
        error: 'Servicio no encontrado o acceso denegado'
      }, { status: 404 });
    }

    // Error genérico
    return NextResponse.json({
      success: false,
      error: error.message || 'Error interno del servidor al eliminar el servicio'
    }, { status: 500 });
  }
}
```

Mejoras:
- ✅ Logging detallado en cada paso del proceso
- ✅ Manejo específico del error `APPOINTMENTS_EXIST`
- ✅ Manejo del error P2003 como fallback
- ✅ Mensajes en español claros y descriptivos
- ✅ Respuestas estructuradas con `details` que incluyen:
- `reason`: Código de la razón del error
- `appointmentCount`: Número de citas asociadas
- `suggestion`: Sugerencia de acción alternativa
- ✅ Códigos HTTP apropiados (400 para validación, 404 para no encontrado, 500 para errores internos)

📊 Flujo de Validación



🎯 Casos de Uso

Caso 1: Servicio sin Citas (Eliminación Exitosa)

Request:

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

Response:

```
{
  "success": true,
  "message": "Servicio eliminado exitosamente"
}
```

Status: 200 OK

Caso 2: Servicio con Citas (Eliminación Bloqueada)

Request:

```
DELETE /api/services/clxyz456
Authorization: Bearer {token}
```

Response:

```
{
  "success": false,
  "error": "No se puede eliminar el servicio porque tiene 5 cita(s) asociada(s)",
  "details": {
    "reason": "APPOINTMENTS_EXIST",
    "appointmentCount": 5,
    "suggestion": "Puede desactivar el servicio en lugar de eliminarlo para mantener
el historial de citas"
  }
}
```

Status: 400 Bad Request

Logs del servidor:

```
[ServiceManager] Checking appointments for service clxyz456: 5 found
[Services API] Cannot delete service - 5 appointments exist
```

Caso 3: Error P2003 (Fallback)

Si por alguna razón la validación previa falla y se intenta eliminar directamente:

Response:

```
{
  "success": false,
  "error": "No se puede eliminar el servicio porque tiene registros asociados (citas, ventas, etc.)",
  "details": {
    "reason": "FOREIGN_KEY_CONSTRAINT",
    "suggestion": "Puede desactivar el servicio en lugar de eliminarlo para mantener la integridad de los datos"
  }
}
```

Status: 400 Bad Request

🔍 Integridad Referencial

Relaciones del Modelo Service

```
model Service {
  // ... campos básicos

  // Relaciones que pueden bloquear la eliminación:
  appointments Appointment[]  // ← Principal causa del error P2003
  serviceUsers ServiceUser[]  // Profesionales asignados
  saleItems    SaleItem[]     // Items de ventas
}
```

Restricciones de Clave Foránea

1. **appointments_serviceId_fkey**
   - Tabla: `appointments`
   - Campo: `serviceId`
   - Referencia: `services.id`
   - Acción: Bloquea eliminación si hay citas

2. **service_users_serviceId_fkey**
   - Tabla: `service_users`
   - Campo: `serviceId`
   - Referencia: `services.id`
   - Acción: CASCADE (se eliminan automáticamente)

3. **sale_items_serviceId_fkey**
   - Tabla: `sale_items`
   - Campo: `serviceId`
   - Referencia: `services.id`
   - Acción: Puede bloquear eliminación

💡 Recomendaciones para el Usuario

Opción 1: Desactivar en lugar de Eliminar (Recomendado)

En lugar de eliminar el servicio, se puede desactivar:

```
PUT /api/services/[id]
Content-Type: application/json

{
  "isActive": false
}
```

Ventajas:
- ✅ Mantiene el historial de citas
- ✅ Preserva la integridad de los datos
- ✅ Permite reactivar el servicio en el futuro
- ✅ No afecta reportes históricos

Opción 2: Eliminar Citas Primero (No Recomendado)

Si realmente se necesita eliminar el servicio:

1. Eliminar o reasignar todas las citas asociadas
2. Luego eliminar el servicio

Desventajas:
- ❌ Pérdida de historial
- ❌ Afecta reportes y estadísticas
- ❌ No reversible

🧪 Testing

Pruebas Manuales Recomendadas

1. **Crear un servicio de prueba**
   ```
   http
     POST /api/services
     {
       "name": "Servicio Test",
       "price": 100,
       "duration": 60
     }
   ```

2. **Crear una cita con ese servicio**
   ```
   http
     POST /api/appointments
     {
       "serviceId": "{serviceId}",
       "clientId": "{clientId}",
       "date": "2025-10-15T10:00:00Z"
     }
   ```

3. **Intentar eliminar el servicio**
   ```
   http
     DELETE /api/services/{serviceId}
   ```

Resultado esperado: Error 400 con mensaje descriptivo

1. **Eliminar la cita**
   `http`
   `DELETE /api/appointments/{appointmentId}`

2. **Intentar eliminar el servicio nuevamente**
   `http`
   `DELETE /api/services/{serviceId}`

Resultado esperado: 200 OK, servicio eliminado

📝 Logs de Debugging

Logs Exitosos (Sin Citas)

```
[Services API] DELETE request received for ID: clxyz123
[Services API] Attempting to delete service: { serviceId: 'clxyz123', tenantId: 'ten-
ant_abc' }
[ServiceManager] Checking appointments for service clxyz123: 0 found
[Services API] Service deleted successfully: clxyz123
```

Logs con Citas (Bloqueado)

```
[Services API] DELETE request received for ID: clxyz456
[Services API] Attempting to delete service: { serviceId: 'clxyz456', tenantId: 'ten-
ant_abc' }
[ServiceManager] Checking appointments for service clxyz456: 5 found
[Services API] DELETE error: Error: APPOINTMENTS_EXIST:5
[Services API] Cannot delete service - 5 appointments exist
```

Logs de Error P2003 (Fallback)

```
[Services API] DELETE request received for ID: clxyz789
[Services API] Attempting to delete service: { serviceId: 'clxyz789', tenantId: 'ten-
ant_abc' }
[ServiceManager] Checking appointments for service clxyz789: 0 found
[Services API] DELETE error: PrismaClientKnownRequestError: Foreign key constraint fai
led
[Services API] Error details: { message: '...', code: 'P2003', meta: {...} }
[Services API] Foreign key constraint violation detected
```

🚀 Deployment

Archivos Modificados

- `app/lib/services/serviceManager.ts`
- `app/app/api/services/[id]/route.ts`

Migraciones de Base de Datos

❌ **No se requieren migraciones**

Breaking Changes

❌ **No hay breaking changes**

Compatibilidad

✅ Compatible con versión actual
✅ No afecta funcionalidad existente
✅ Solo mejora el manejo de errores

📚 Referencias

Documentación Prisma

- Error P2003 - Foreign Key Constraint (https://www.prisma.io/docs/reference/api-reference/error-reference#p2003)
- Referential Actions (https://www.prisma.io/docs/concepts/components/prisma-schema/relations/referential-actions)

Código Relacionado

- `app/prisma/schema.prisma` - Definición de modelos y relaciones
- `app/lib/services/serviceManager.ts` - Lógica de negocio de servicios
- `app/app/api/services/[id]/route.ts` - Endpoint REST de servicios

✅ Checklist de Implementación

- [x] Implementar validación en `serviceManager.deleteService()`
- [x] Agregar logging detallado
- [x] Manejar error `APPOINTMENTS_EXIST` en endpoint
- [x] Manejar error P2003 como fallback
- [x] Mensajes de error en español
- [x] Incluir sugerencias en respuestas de error
- [x] Documentación técnica completa
- [ ] Testing manual
- [ ] Code review
- [ ] Merge a main
- [ ] Deployment a producción
- [ ] Verificación en producción

🎓 Lecciones Aprendidas

1. **Validación Previa es Clave:** Siempre validar restricciones de integridad referencial antes de intentar operaciones de eliminación.
2. **Mensajes Claros:** Los mensajes de error deben ser descriptivos y en el idioma del usuario, con sugerencias de acción.
3. **Logging Detallado:** El logging exhaustivo facilita el debugging en producción.
4. **Alternativas al Usuario:** Ofrecer alternativas (como desactivar) en lugar de solo bloquear la acción.
5. **Manejo de Errores en Capas:** Validar en la capa de servicio y manejar errores específicos en la capa de API.

📞 Contacto y Soporte

Para preguntas o problemas relacionados con este fix:
- Revisar logs del servidor con `[Services API]` y `[ServiceManager]`
- Verificar que el servicio no tenga citas asociadas
- Considerar desactivar en lugar de eliminar

Versión del Documento: 1.0
Última Actualización: 13 de Octubre, 2025
Autor: CitaPlanner Development Team