

# Fase 4: Vista de Calendario por Profesional

---

**Versión:** 1.8.0

**Fecha:** Octubre 14, 2025

**PR:** #103

**Estado:** Implementada

---

## Tabla de Contenidos

---

1. [Resumen Ejecutivo](#)
  2. [Arquitectura](#)
  3. [Funcionalidades Implementadas](#)
  4. [API Endpoints](#)
  5. [Componentes Frontend](#)
  6. [Servicios de Negocio](#)
  7. [Tipos TypeScript](#)
  8. [Integración con Fases Anteriores](#)
  9. [Permisos y Seguridad](#)
  10. [Guía de Uso](#)
  11. [Testing](#)
  12. [Próximos Pasos](#)
- 

## Resumen Ejecutivo

---

La Fase 4 introduce un **sistema completo de calendario visual por profesional** que permite:

- **Visualización interactiva** de citas en vistas mensual, semanal, diaria y agenda
- **Drag & Drop** para reprogramar citas con validaciones automáticas
- **Creación rápida** de citas directamente desde el calendario
- **Gestión completa** de disponibilidad basada en horarios (Fase 1) y asignaciones (Fase 2)
- **Filtros avanzados** por sucursal, estado, servicio y profesional
- **Permisos granulares** según rol de usuario

### Beneficios Clave

- ✓ **UX Superior:** Interfaz intuitiva con arrastrar y soltar
  - ✓ **Validaciones Robustas:** Previene conflictos de horarios automáticamente
  - ✓ **Integración Total:** Usa horarios, excepciones y asignaciones de fases anteriores
  - ✓ **Responsive:** Funciona perfectamente en desktop y móvil
  - ✓ **Rendimiento:** Carga eficiente de datos con paginación
  - ✓ **Seguridad:** Permisos estrictos según rol de usuario
-

## Arquitectura

### Stack Tecnológico

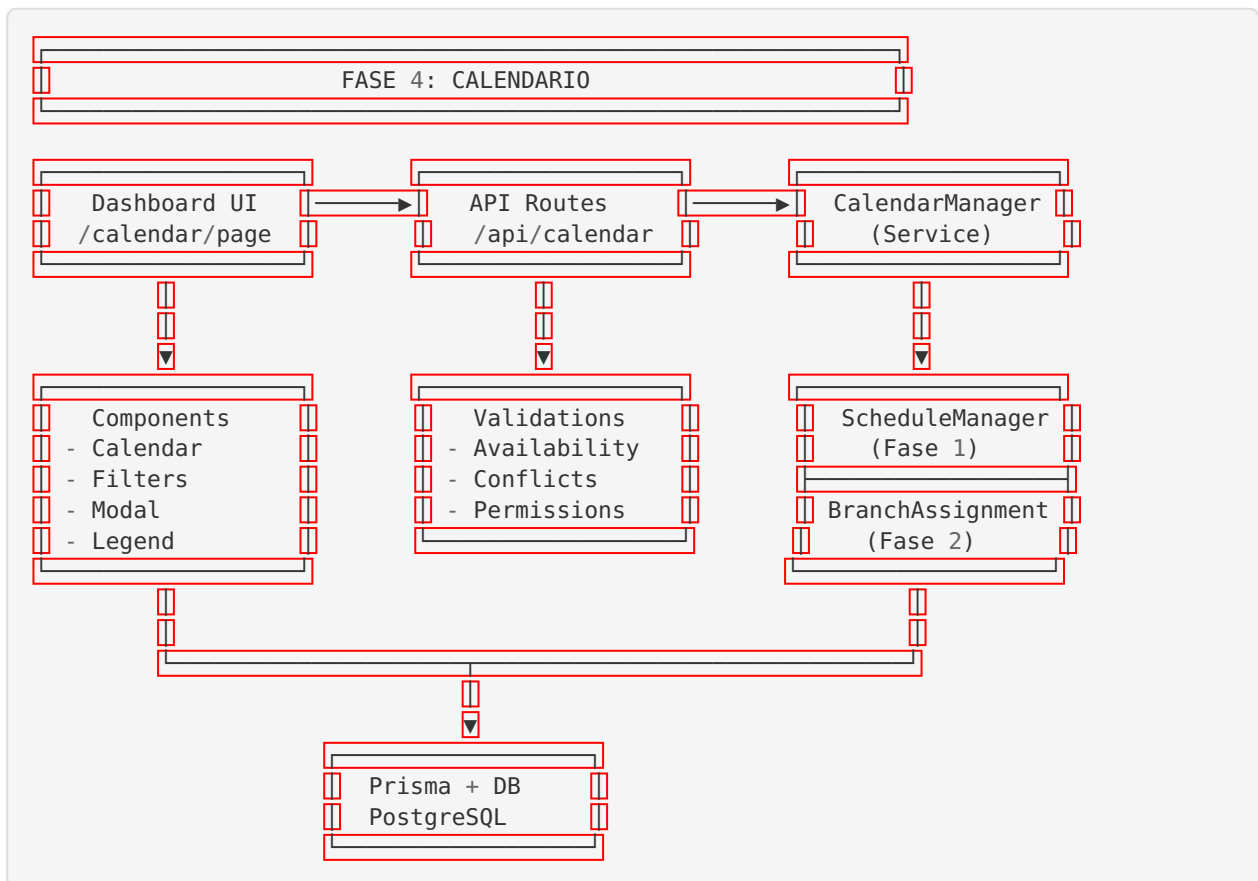
#### Frontend:

- React 18 (Client Components)
- **Next.js** 14 (App Router)
- react-big-calendar 1.x
- date-fns 3.x
- TailwindCSS 3.x
- TypeScript 5.x

#### Backend:

- **Next.js** API Routes
- Prisma ORM
- PostgreSQL
- NextAuth (autenticación)

### Diagrama de Arquitectura



## Funcionalidades Implementadas

### 1. Vistas de Calendario

#### Vista Mensual

- Muestra todo el mes con citas en formato compacto
- Indicador visual de día actual

- Colores diferenciados por estado de cita
- Popup con detalles al hacer hover

### **Vista Semanal**

- Muestra 7 días con slots de tiempo cada 30 minutos
- Horario de trabajo de 8:00 AM a 8:00 PM (configurable)
- Drag & drop habilitado
- Visualización de bloques de disponibilidad

### **Vista Diaria**

- Vista detallada de un solo día
- Slots de tiempo precisos
- Ideal para gestión intensiva de citas
- Muestra todas las citas del día con detalles completos

### **Vista Agenda**

- Lista cronológica de citas
- Útil para vista general de próximas citas
- Filtros aplicables

## **2. Gestión de Citas**

### **Crear Cita**

1. Click en slot de tiempo disponible
2. Se abre modal con formulario
3. Seleccionar cliente, servicio, sucursal
4. Validación automática de disponibilidad
5. Creación instantánea

### **Editar Cita**

1. Click en cita existente
2. Modal con datos prellenados
3. Modificar campos necesarios
4. Guardar cambios

### **Cancelar Cita**

1. Abrir modal de cita
2. Botón "Cancelar Cita"
3. Confirmación requerida
4. Estado actualizado a CANCELLED

### **Reprogramar Cita (Drag & Drop)**

1. Arrastrar cita a nuevo horario
2. Validación automática de disponibilidad
3. Si es válido: actualización instantánea
4. Si no: revertir cambio y mostrar error

### 3. Filtros Avanzados





```
interface CalendarFilters {
  professionalId?: string; // Selector de profesional (admin/gerente)
  branchId?: string; // Filtrar por sucursal
  status?: AppointmentStatus; // Filtrar por estado
  serviceId?: string; // Filtrar por servicio
  startDate: Date; // Rango de fechas
  endDate: Date;
}
```

#### Características:

- Aplicación en tiempo real sin recargar
- Múltiples filtros combinables
- Opciones dinámicas según permisos del usuario
- Estado persistente durante la sesión

### 4. Visualización de Disponibilidad

El calendario muestra visualmente:

-  **Bloques Disponibles:** Fondo blanco, clickeable
-  **Bloques No Disponibles:** Fondo gris, no clickeable
-  **Excepciones (Vacaciones):** Patrón diferenciado
-  **Override por Sucursal:** Color distintivo si aplica

```
interface AvailabilityBlock {
  id: string;
  start: Date;
  end: Date;
  isAvailable: boolean;
  type: 'regular' | 'exception' | 'override';
  reason?: string;
  branchId?: string;
}
```

### 5. Validaciones Automáticas

El sistema valida automáticamente:

1. **Horario dentro de disponibilidad:** No se pueden crear citas fuera de horarios de trabajo
2. **Sin solapamientos:** No permite citas que se solapen con otras existentes
3. **Respeto a excepciones:** Bloquea citas durante vacaciones/bajas médicas
4. **Duración correcta:** Valida que la cita termine dentro del mismo bloque disponible
5. **Permisos:** Solo usuarios autorizados pueden ver/modificar calendarios



## API Endpoints

### GET /api/calendar/professional/[id]

Obtiene eventos del calendario de un profesional.

#### Query Parameters:

```
{
  startDate: string;      // ISO date (required)
  endDate: string;        // ISO date (required)
  branchId?: string;      // Filtrar por sucursal
  status?: string;        // Filtrar por estado
  serviceId?: string;     // Filtrar por servicio
}
```

**Response:**

```
{
  success: boolean;
  events: CalendarEvent[];
  availability: ProfessionalAvailability;
}
```

**Ejemplo:**

```
GET /api/calendar/professional/prof123?
startDate=2025-10-01&endDate=2025-10-31&branchId=branch456
```

**GET /api/calendar/availability/[professionalId]**

Obtiene disponibilidad de un profesional (horarios y excepciones).

**Query Parameters:**

```
{
  startDate: string;      // ISO date (required)
  endDate: string;        // ISO date (required)
  branchId?: string;      // Considerar override de sucursal
}
```

**Response:**

```
{
  success: boolean;
  availability: {
    professionalId: string;
    startDate: Date;
    endDate: Date;
    blocks: AvailabilityBlock[];
    exceptions: ScheduleExceptionInfo[];
  }
}
```

**GET /api/calendar/availability/[professionalId]/slots**

Obtiene slots disponibles para agendar en un día específico.

**Query Parameters:**

```
{
  date: string;           // ISO date (required)
  duration: number;       // Duración en minutos (required)
  branchId?: string;      // Considerar override de sucursal
}
```

**Response:**

```
{
  success: boolean;
  slots: Date[];         // Array de horarios disponibles
}
```

**Ejemplo:**

```
GET /api/calendar/availability/prof123/slots?date=2025-10-15&duration=60
```

**POST /api/calendar/availability/validate**

Valida si se puede crear/mover una cita en un horario específico.

**Request Body:**

```
{
  professionalId: string;
  startTime: string;      // ISO date
  endTime: string;       // ISO date
  branchId?: string;
  excludeAppointmentId?: string; // Para edición
}
```

**Response:**

```
{
  success: boolean;
  validation: {
    isValid: boolean;
    reason?: string;
    conflictingAppointments?: string[];
    availabilityIssues?: string[];
  }
}
```

**POST /api/calendar/appointments**

Crea una nueva cita desde el calendario.

**Request Body:**

```
{
  professionalId: string;
  clientId: string;
  serviceId: string;
  branchId: string;
  startTime: string;           // ISO date
  endTime: string;            // ISO date
  notes?: string;
}
```

**Response:**

```
{
  success: boolean;
  message: string;
  appointment: Appointment;
}
```

**PATCH /api/calendar/appointments/[id]/reschedule**

Reprograma una cita (drag & drop).

**Request Body:**

```
{
  newStartTime: string;        // ISO date
  newEndTime: string;          // ISO date
  reason?: string;
}
```

**Response:**

```
{
  success: boolean;
  message: string;
  appointment: Appointment;
}
```

**GET /api/calendar/statistics/[professionalId]**

Obtiene estadísticas del calendario de un profesional.

**Query Parameters:**

```
{
  startDate: string;           // ISO date (required)
  endDate: string;             // ISO date (required)
}
```

**Response:**

```
{
  success: boolean;
  statistics: {
    professionalId: string;
    period: { start: Date; end: Date };
    totalAppointments: number;
    appointmentsByStatus: {
      PENDING: number;
      CONFIRMED: number;
      IN_PROGRESS: number;
      COMPLETED: number;
      CANCELLED: number;
      NO_SHOW: number;
    };
    utilizationRate: number; // % de tiempo ocupado
    averageAppointmentDuration: number; // minutos
    peakHours: Array<{ hour: number; count: number }>;
    peakDays: Array<{ day: string; count: number }>;
  }
}
```

## GET /api/professionals/me

Obtiene los datos del profesional del usuario autenticado.

### Response:

```
{
  success: boolean;
  professional: {
    id: string;
    userId: string;
    specialization: string;
    user: {
      id: string;
      firstName: string;
      lastName: string;
      email: string;
    };
  }
}
```



## Componentes Frontend

### ProfessionalCalendar

**Ubicación:** app/components/calendar/ProfessionalCalendar.tsx

Componente principal que integra react-big-calendar.

### Props:



```
interface ProfessionalCalendarProps {
  events: CalendarEvent[];
  view: CalendarView;
  date: Date;
  onNavigate: (newDate: Date) => void;
  onView: (newView: CalendarView) => void;
  onSelectEvent?: (event: CalendarEvent) => void;
  onSelectSlot?: (slotInfo: { start: Date; end: Date }) => void;
  onEventDrop?: (data: { event: CalendarEvent; start: Date; end: Date }) => void;
  onEventResize?: (data: { event: CalendarEvent; start: Date; end: Date }) => void;
  availabilityBlocks?: AvailabilityBlock[];
  loading?: boolean;
}
```

#### Características:

- Drag & drop habilitado
- Resizable events
- Custom event styling por estado
- Slot styling según disponibilidad
- Localización en español
- Responsive design

## CalendarFilters

**Ubicación:** app/components/calendar/CalendarFilters.tsx

Filtros y controles del calendario.

#### Props:

```
interface CalendarFiltersProps {
  view: CalendarView;
  onViewChange: (view: CalendarView) => void;
  selectedBranchId?: string;
  onBranchChange: (branchId: string) => void;
  selectedStatus?: AppointmentStatus | 'ALL';
  onStatusChange: (status: AppointmentStatus | 'ALL') => void;
  selectedServiceId?: string;
  onServiceChange: (serviceId: string) => void;
  selectedProfessionalId?: string;
  onProfessionalChange?: (professionalId: string) => void;
  filterOptions: FilterOptions;
  showProfessionalSelector?: boolean;
}
```

#### Características:

- Selector de vista (mes/semana/día/agenda)
- Filtros dinámicos según permisos
- Aplicación en tiempo real
- Diseño responsive

## CalendarLegend

**Ubicación:** app/components/calendar/CalendarLegend.tsx

Leyenda de colores y estados.

### Características:

- Muestra colores de estados de cita
- Indica disponibilidad
- Diseño compacto
- Fácil referencia visual

## AppointmentModal

**Ubicación:** app/components/calendar/AppointmentModal.tsx

Modal para crear/editar/ver citas.

### Props:

```
interface AppointmentModalProps {
  isOpen: boolean;
  onClose: () => void;
  onSave: (data: AppointmentFormData) => Promise<void>;
  onCancel?: (appointmentId: string) => Promise<void>;
  appointment?: CalendarEvent;
  professionalId: string;
  initialStartTime?: Date;
  initialEndTime?: Date;
  clients: Array<{ id: string; name: string }>;
  services: Array<{ id: string; name: string; duration: number }>;
  branches: Array<{ id: string; name: string }>;
  mode: 'create' | 'edit' | 'view';
}
```

### Características:

- Tres modos: crear, editar, ver
- Validación de formulario
- Auto-cálculo de endTime basado en servicio
- Botón de cancelar cita
- Manejo de errores inline



## Servicios de Negocio

### CalendarManager

**Ubicación:** app/lib/services/calendarManager.ts

Servicio central para gestión de calendario.

## Métodos Principales

```
class CalendarManager {
  // Obtiene eventos del calendario con filtros
  static async getCalendarEvents(
    filters: CalendarFilters,
    requestingUserId: string,
    requestingUserRole: string
  ): Promise<{ events: CalendarEvent[]; availability: ProfessionalAvailability }>;

  // Obtiene disponibilidad de un profesional
  static async getProfessionalAvailability(
    professionalId: string,
    startDate: Date,
    endDate: Date,
    branchId?: string
  ): Promise<ProfessionalAvailability>;

  // Valida si se puede crear/mover una cita
  static async validateAvailability(
    options: ValidateAvailabilityOptions
  ): Promise<AvailabilityValidation>;

  // Obtiene estadísticas del calendario
  static async getCalendarStatistics(
    professionalId: string,
    startDate: Date,
    endDate: Date
  ): Promise<CalendarStatistics>;

  // Obtiene slots disponibles para agendar
  static async getAvailableSlots(
    professionalId: string,
    date: Date,
    serviceDuration: number,
    branchId?: string
  ): Promise<Date[]>;

  // Valida acceso al calendario (privado)
  private static async validateCalendarAccess(
    professionalId: string,
    requestingUserId: string,
    requestingUserRole: string
  ): Promise<void>;
}
```



## Tipos TypeScript

**Archivo:** `app/lib/types/calendar.ts`

Contiene todos los tipos relacionados con el calendario:

## Tipos Principales

```
// Evento del calendario
interface CalendarEvent extends BigCalendarEvent {
  id: string;
  title: string;
  start: Date;
  end: Date;
  resource?: CalendarEventResource;
  allDay?: boolean;
}

// Recurso del evento (datos de la cita)
interface CalendarEventResource {
  appointmentId: string;
  professionalId: string;
  professionalName: string;
  clientId: string;
  clientName: string;
  serviceId: string;
  serviceName: string;
  branchId: string;
  branchName: string;
  status: AppointmentStatus;
  notes?: string;
  price?: number;
  duration?: number;
}

// Bloque de disponibilidad
interface AvailabilityBlock {
  id: string;
  start: Date;
  end: Date;
  isAvailable: boolean;
  type: 'regular' | 'exception' | 'override';
  reason?: string;
  branchId?: string;
}

// Filtros del calendario
interface CalendarFilters {
  professionalId?: string;
  branchId?: string;
  status?: AppointmentStatus | 'ALL';
  serviceId?: string;
  startDate: Date;
  endDate: Date;
}

// Vistas del calendario
type CalendarView = 'month' | 'week' | 'day' | 'agenda';
```

## Helpers Disponibles

```
// Crear evento desde appointment
function createCalendarEventFromAppointment(appointment: any): CalendarEvent;

// Obtener color según estado
function getStatusColor(status: AppointmentStatus): string;

// Validar si está en horario laboral
function isWithinWorkingHours(
  date: Date,
  workingHours: { start: string; end: string }
): boolean;

// Obtener rango de fechas para una vista
function getDateRangeForView(
  date: Date,
  view: CalendarView
): { start: Date; end: Date };
```

## Integración con Fases Anteriores

### Fase 1: Sistema de Horarios

#### Integración:

- CalendarManager usa `scheduleManager.ts` para obtener horarios
- Respeta `ProfessionalSchedule` con `dayOfWeek`, `startTime`, `endTime`
- Procesa `ScheduleException` para bloquear fechas
- Calcula disponibilidad basada en configuración de horarios

#### Ejemplo:

```
// Obtener horarios del profesional (Fase 1)
const professional = await prisma.professional.findUnique({
  where: { id: professionalId },
  include: {
    schedules: true, // ← Horarios de Fase 1
    scheduleExceptions: true, // ← Excepciones de Fase 1
  },
});

// Construir bloques de disponibilidad
for (const day of days) {
  const schedule = professional.schedules.find(
    s => s.dayOfWeek === dayOfWeek
  );

  if (schedule && schedule.isAvailable) {
    blocks.push({
      start: scheduleStartTime,
      end: scheduleEndTime,
      isAvailable: true,
      type: 'regular',
    });
  }
}
```

---

## Fase 2: Asignaciones de Sucursales

### Integración:

- CalendarManager considera `branchAssignments` con sucursal primaria
- Aplica `scheduleOverride` cuando está definido
- Filtra por sucursal en queries
- Valida permisos de gerente según sucursales asignadas

### Ejemplo:

```
// Verificar override por sucursal (Fase 2)
if (branchId) {
  const assignment = professional.branchAssignments.find(
    a => a.branchId === branchId && a.scheduleOverride
  );

  if (assignment && assignment.scheduleOverride) {
    // Aplicar horarios override en lugar de horarios regulares
    const override = assignment.scheduleOverride as any;
    const overrideDay = override[dayOfWeek.toLowerCase()];

    if (overrideDay && overrideDay.isAvailable) {
      blocks.push({
        start: overrideStartTime,
        end: overrideEndTime,
        isAvailable: true,
        type: 'override',
        branchId,
      });
    }
  }
}
```

---

## Fase 3: Reportes

### Integración Futura:

- Las estadísticas del calendario complementan los reportes de Fase 3
  - `CalendarStatistics` puede ser usado por `reportManager.ts`
  - Métricas de utilización alimentan dashboards
-

## Permisos y Seguridad

### Matriz de Permisos

Rol	Ver Pro- pio Calen- dario	Ver Otros Calen- darios	Crear Citas	Editar Citas	Cancelar Citas	Repro- gramar
PROFES- SIONAL	✓	✗	✓	✓	✓	✓
MANAGER	✓	✓ (sucurs- ales)	✓	✓	✓	✓
ADMIN	✓	✓	✓	✓	✓	✓
SU- PER_ADMI N	✓	✓	✓	✓	✓	✓
CLIENT	✗	✗	✗	✗	✗	✗

## Validación de Permisos

```
// En CalendarManager.validateCalendarAccess()
private static async validateCalendarAccess(
  professionalId: string,
  requestingUserId: string,
  requestingUserRole: string
): Promise<void> {
  // Admin puede ver todo
  if (requestingUserRole === 'ADMIN' || requestingUserRole === 'SUPER_ADMIN') {
    return;
  }

  // Profesional solo puede ver su propio calendario
  if (requestingUserRole === 'PROFESSIONAL') {
    const professional = await prisma.professional.findFirst({
      where: {
        userId: requestingUserId,
        id: professionalId,
      },
    });

    if (!professional) {
      throw new Error('No tienes permiso para ver este calendario');
    }
    return;
  }

  // Gerente puede ver calendarios de profesionales de sus sucursales
  if (requestingUserRole === 'MANAGER') {
    // ... validación de sucursales
  }

  throw new Error('No tienes permiso para acceder a calendarios');
}
```

---

## Guía de Uso

### Para Profesionales

#### 1. Acceder al Calendario:

- Ir a `/dashboard/calendar`
- El calendario muestra automáticamente tus citas

#### 2. Crear una Cita:

- Click en un slot de tiempo disponible
- Rellenar formulario (cliente, servicio, sucursal)
- Guardar

#### 3. Reprogramar una Cita:

- Arrastrar la cita a nuevo horario
- Sistema valida automáticamente
- Si es válido, se guarda instantáneamente

#### 4. Editar una Cita:

- Click en la cita existente



- Modificar campos necesarios
- Guardar cambios

#### 5. **Cancelar una Cita:**

- Click en la cita
- Botón "Cancelar Cita"
- Confirmar acción

#### 6. **Cambiar Vista:**

- Usar botones Mes / Semana / Día / Agenda
- Navegar con botones Anterior / Siguiente

#### 7. **Aplicar Filtros:**

- Seleccionar sucursal para filtrar
- Seleccionar estado de citas
- Seleccionar servicio específico

## Para Administradores/Gerentes

#### 1. **Ver Calendario de un Profesional:**

- Ir a `/dashboard/calendar`
- Seleccionar profesional del dropdown
- Ver su calendario completo

#### 2. **Crear Cita para un Profesional:**

- Seleccionar profesional
- Click en slot disponible
- Crear cita normalmente

#### 3. **Gestionar Múltiples Sucursales:**

- Filtrar por sucursal específica
- Ver horarios override si aplican
- Considerar disponibilidad por sucursal

#### 4. **Análisis de Utilización:**





- Usar vista de estadísticas
- Ver tasa de ocupación
- Identificar horas/días pico



## Testing

### Testing Manual Recomendado

#### 1. **Creación de Citas**

- [ ] Crear cita en horario disponible →  Se crea correctamente
- [ ] Crear cita fuera de horario →  Error: "Horario no disponible"
- [ ] Crear cita en horario ocupado →  Error: "Ya existe una cita"
- [ ] Crear cita durante excepción →  Error: "Profesional no disponible"

## 2. Drag & Drop

- [ ] Arrastrar cita a horario válido → ☒ Se mueve correctamente
- [ ] Arrastrar cita a horario inválido → ☒ Se revierte el cambio
- [ ] Arrastrar cita que se solapa → ☒ Error de validación

## 3. Filtros

- [ ] Filtrar por sucursal → ☒ Solo muestra citas de esa sucursal
- [ ] Filtrar por estado → ☒ Solo muestra citas con ese estado
- [ ] Combinar filtros → ☒ Aplica ambos correctamente
- [ ] Limpiar filtros → ☒ Muestra todas las citas

## 4. Permisos

- [ ] Profesional ve solo su calendario → ☒ Correcto
- [ ] Gerente ve calendarios de su sucursal → ☒ Correcto
- [ ] Admin ve todos los calendarios → ☒ Correcto
- [ ] Cliente no tiene acceso → ☒ Redirección o error 403

## 5. Vistas

- [ ] Vista mensual funciona → ☒ Muestra mes completo
- [ ] Vista semanal funciona → ☒ Muestra semana con slots
- [ ] Vista diaria funciona → ☒ Muestra día detallado
- [ ] Vista agenda funciona → ☒ Muestra lista de citas

## 6. Disponibilidad

- [ ] Horarios regulares se muestran → ☒ Bloques blancos
- [ ] Horarios no disponibles bloqueados → ☒ Bloques grises
- [ ] Excepciones se respetan → ☒ No se pueden crear citas
- [ ] Override por sucursal funciona → ☒ Usa horarios override



## Próximos Pasos

### Mejoras Futuras

#### 1. Notificaciones en Tiempo Real

- WebSockets para actualizaciones live
- Notificación cuando otro usuario mueve/crea citas

#### 2. Exportar Calendario

- Exportar a iCal/Google Calendar
- Sincronización bidireccional

#### 3. Vista de Múltiples Profesionales

- Vista de equipo completo
- Comparación side-by-side

#### 4. Recordatorios Automáticos

- Email/SMS antes de cita
- Integración con sistema de notificaciones

## 5. Citas Recurrentes

- Crear series de citas
- Gestión de recurrencia

## 6. Mobile App

- App nativa iOS/Android
- Notificaciones push



## Referencias

### Documentación Externa

- [react-big-calendar](https://github.com/jquense/react-big-calendar) (<https://github.com/jquense/react-big-calendar>)
- [date-fns](https://date-fns.org/) (<https://date-fns.org/>)
- [Next.js API Routes](https://nextjs.org/docs/app/building-your-application/routing/route-handlers) (<https://nextjs.org/docs/app/building-your-application/routing/route-handlers>)
- [Prisma ORM](https://www.prisma.io/docs) (<https://www.prisma.io/docs>)

### Documentación Interna

- [FASE1\\_SCHEDULE\\_MANAGEMENT.md](#) ([./FASE1\\_SCHEDULE\\_MANAGEMENT.md](#))
- [FASE2\\_MASS\\_ASSIGNMENT.md](#) ([./FASE2\\_MASS\\_ASSIGNMENT.md](#))
- [FASE3\\_REPORTS.md](#) ([./FASE3\\_REPORTS.md](#))
- [CHANGELOG.md](#) ([../CHANGELOG.md](#))



## Soporte y Contacto

Para preguntas o soporte:

- Revisar documentación de fases anteriores
- Consultar CHANGELOG.md para historial de cambios
- Revisar código fuente con comentarios detallados

---

**Documento generado:** Octubre 14, 2025

**Última actualización:** v1.8.0

**Autor:** CitaPlanner Development Team