📅 Fase 1: Gestión de Horarios Detallados por Profesional

Fecha de implementación: 14 de Octubre, 2025
Versión: 1.4.0 → 1.5.0
Estado: ✅ Implementado

📋 Resumen Ejecutivo

Esta fase implementa un sistema completo de gestión de horarios detallados para profesionales en CitaPlanner. Permite configurar horarios semanales personalizados, múltiples bloques de tiempo por día, y gestionar excepciones como vacaciones o días especiales.

🎯 Objetivos Cumplidos

- ✅ Sistema de horarios por día de la semana
- ✅ Múltiples bloques de tiempo por día (ej: mañana y tarde)
- ✅ Gestión de excepciones (vacaciones, días especiales, festivos)
- ✅ Validación de disponibilidad basada en horarios
- ✅ Interfaz visual intuitiva para configuración
- ✅ Cálculo automático de slots disponibles
- ✅ Estadísticas de horarios

🏗️ Arquitectura Implementada

1. Tipos y Estructuras de Datos

Archivo: `app/lib/types/schedule.ts`

```
// Enums principales
enum DayOfWeek { MONDAY, TUESDAY, ... }
enum ExceptionType { VACATION, SICK_LEAVE, SPECIAL_DAY, HOLIDAY,
        CUSTOM }

// Estructuras de datos
interface TimeBlock {
  startTime: string;  // HH:mm
  endTime: string;    // HH:mm
}

interface DaySchedule {
```

```
  day: DayOfWeek;
  isWorkingDay: boolean;
  timeBlocks: TimeBlock[];
}

interface ScheduleException {
  id?: string;
  date: string;           // YYYY-MM-DD
  type: ExceptionType;
  reason?: string;
  isAvailable: boolean;
  timeBlocks?: TimeBlock[];
}

interface ScheduleConfig {
  version: string;
  defaultSchedule: DaySchedule[];
  exceptions: ScheduleException[];
  timezone: string;
  lastUpdated: string;
  updatedBy?: string;
}
```

2. Servicio de Gestión de Horarios

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

Clase: `ScheduleManager`

Métodos Principales:

1. **createDefaultConfig(userId?: string): ScheduleConfig**
   - Crea configuración por defecto (Lun-Vie 9:00-18:00)
2. **validateScheduleConfig(config: ScheduleConfig): ScheduleValidationResult**
   - Valida configuración completa
   - Verifica formato de horas
   - Detecta solapamientos
   - Valida excepciones
3. **getScheduleForDate(config: ScheduleConfig, date: Date): DaySchedule | null**
   - Obtiene horario para fecha específica
   - Considera excepciones
4. **calculateAvailableSlots(config, query, appointments): AvailableSlot[]**
   - Calcula slots disponibles
   - Considera citas existentes
   - Respeta horarios y excepciones

5. **isAvailable(config, date, startTime, endTime, appointments): boolean**
   - Verifica disponibilidad en horario específico
6. **calculateStats(config: ScheduleConfig): ScheduleStats**
   - Calcula estadísticas del horario
   - Horas semanales, días laborables, etc.
7. **addException(config, exception, userId?): ScheduleConfig**
   - Agrega excepción al horario
8. **removeException(config, exceptionId, userId?): ScheduleConfig**
   - Elimina excepción
9. **updateDaySchedule(config, day, schedule, userId?): ScheduleConfig**
   - Actualiza horario de un día específico

3. API Endpoints

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

GET `/api/professionals/[id]/schedule`

Obtiene el horario de un profesional

Response:

```
{
  "success": true,
  "data": {
    "professionalId": "prof_123",
    "professionalName": "Dr. Juan Pérez",
    "schedule": {
      "version": "1.0.0",
      "defaultSchedule": [...],
      "exceptions": [...],
      "timezone": "America/Mexico_City",
      "lastUpdated": "2025-10-14T10:00:00Z"
    },
    "stats": {
      "totalWorkingHours": 45,
      "workingDays": 5,
      "averageHoursPerDay": 9,
      "totalExceptions": 3,
      "upcomingExceptions": 2
    }
  }
}
```

PUT `/api/professionals/[id]/schedule`

Actualiza el horario completo

Request:

```
{
  "schedule": {
    "version": "1.0.0",
    "defaultSchedule": [...],
    "exceptions": [...],
    "timezone": "America/Mexico_City"
  }
}
```

Response:

```
{
  "success": true,
  "message": "Horario actualizado exitosamente",
  "data": {
    "professionalId": "prof_123",
    "schedule": {...},
    "stats": {...}
  }
}
```

POST `/api/professionals/[id]/schedule/exceptions`

Agrega una excepción al horario

Request:

```
{
  "exception": {
    "date": "2025-12-25",
    "type": "HOLIDAY",
    "reason": "Navidad",
    "isAvailable": false
  }
}
```

4. Componentes Frontend

ScheduleManager Component

Archivo: `app/components/ScheduleManager.tsx`

Props:

```
interface ScheduleManagerProps {
  professionalId: string;
```

```
  professionalName: string;
  initialSchedule?: ScheduleConfig;
  onSave: (schedule: ScheduleConfig) => Promise<void>;
  onCancel?: () => void;
}
```

Características: - ✅ Editor visual de horario semanal - ✅ Gestión de múltiples bloques por día - ✅ Selector de horas con validación - ✅ Gestión de excepciones - ✅ Validación en tiempo real - ✅ Mensajes de error descriptivos - ✅ Interfaz responsive

Sub-componentes: 1. **WeeklyScheduleEditor**: Editor de horario semanal 2. **ExceptionsEditor**: Gestor de excepciones

Página de Gestión de Horarios

Archivo: `app/dashboard/professionals/schedule/[id]/page.tsx`

Ruta: `/dashboard/professionals/schedule/[id]`

Funcionalidades: - Carga datos del profesional - Carga horario existente o crea uno por defecto - Maneja guardado de cambios - Navegación con breadcrumbs - Manejo de errores - Loading states

📊 Estructura de Datos en Base de Datos

El horario se almacena en el campo `scheduleConfig` del modelo `Professional` como JSON:

```
model Professional {
  id             String   @id @default(cuid())
  name           String
  email          String?
  phone          String?
  scheduleConfig Json?    // ← Aquí se guarda el horario
  // ... otros campos
}
```

Ejemplo de datos almacenados:

```
{
  "version": "1.0.0",
  "defaultSchedule": [
    {
      "day": "MONDAY",
      "isWorkingDay": true,
      "timeBlocks": [
        { "startTime": "09:00", "endTime": "13:00" },
        { "startTime": "15:00", "endTime": "19:00" }
```

```
        ]
      },
      {
        "day": "TUESDAY",
        "isWorkingDay": true,
        "timeBlocks": [
          { "startTime": "09:00", "endTime": "18:00" }
        ]
      },
      // ... resto de días
    ],
    "exceptions": [
      {
        "id": "exc_1729000000_abc123",
        "date": "2025-12-25",
        "type": "HOLIDAY",
        "reason": "Navidad",
        "isAvailable": false
      }
    ],
    "timezone": "America/Mexico_City",
    "lastUpdated": "2025-10-14T10:00:00Z",
    "updatedBy": "user_123"
}
```

🔧 Validaciones Implementadas

1. Validación de Formato

- ✅ Horas en formato HH:mm (24h)
- ✅ Fechas en formato YYYY-MM-DD
- ✅ Versión del schema
- ✅ Zona horaria válida

2. Validación de Lógica

- ✅ Hora de fin posterior a hora de inicio
- ✅ Duración mínima de 15 minutos
- ✅ Sin solapamientos entre bloques
- ✅ Al menos un día laboral
- ✅ Días laborales con horarios definidos

3. Validación de Excepciones

- ✅ Fecha válida
- ✅ Tipo de excepción válido
- ✅ Bloques de tiempo válidos si está disponible

🎨 Interfaz de Usuario

Vista Principal

```
┌─────────────────────────────────────────────────────┐
│ Gestión de Horarios                                 │
│ Configurar horario de trabajo para Dr. Juan Pérez   │
├─────────────────────────────────────────────────────┤
│ [Horario Semanal] [Excepciones (3)]                 │
├─────────────────────────────────────────────────────┤
│                                                     │
│ ☑ Lunes                          [+ Agregar bloque] │
│   09:00 a 13:00 [🗑]                                │
│   15:00 a 19:00 [🗑]                                │
│                                                     │
│ ☑ Martes                         [+ Agregar bloque] │
│   09:00 a 18:00 [🗑]                                │
│                                                     │
│ ☐ Sábado                                            │
│ ☐ Domingo                                           │
│                                                     │
├─────────────────────────────────────────────────────┤
│                          [Cancelar] [Guardar]       │
└─────────────────────────────────────────────────────┘
```

Vista de Excepciones

```
┌─────────────────────────────────────────────────────┐
│ Excepciones                      [+ Nueva Excepción] │
├─────────────────────────────────────────────────────┤
│ ┌─────────────────────────────────────────────────┐ │
│ │ Fecha: [2025-12-25]  Tipo: [Festivo ▼]          │ │
│ │ ☐ Disponible este día                           │ │
│ │ Motivo: Navidad                                 │ │
│ │ [🗑 Eliminar excepción]                         │ │
│ └─────────────────────────────────────────────────┘ │
└─────────────────────────────────────────────────────┘
```

🚀 Casos de Uso

1. Configurar Horario Regular

```
// Profesional trabaja Lun-Vie 9:00-18:00
const config = ScheduleManager.createDefaultConfig();
// Guardar en BD
await updateProfessional(id, { scheduleConfig: config });
```

2. Agregar Horario Partido

```
// Lunes: 9:00-13:00 y 15:00-19:00
const config = ScheduleManager.updateDaySchedule(
  currentConfig,
  DayOfWeek.MONDAY,
  {
    isWorkingDay: true,
    timeBlocks: [
      { startTime: '09:00', endTime: '13:00' },
      { startTime: '15:00', endTime: '19:00' }
    ]
  }
);
```

3. Agregar Vacaciones

```
const config = ScheduleManager.addException(
  currentConfig,
  {
    date: '2025-12-20',
    type: ExceptionType.VACATION,
    reason: 'Vacaciones de invierno',
    isAvailable: false
  }
);
```

4. Consultar Disponibilidad

```
const slots = ScheduleManager.calculateAvailableSlots(
  config,
  {
    professionalId: 'prof_123',
    startDate: '2025-10-15',
    endDate: '2025-10-20',
    serviceDuration: 60,
    slotInterval: 30
  },
  existingAppointments
);
```

5. Verificar Disponibilidad Específica

```
const isAvailable = ScheduleManager.isAvailable(
  config,
  new Date('2025-10-15'),
  '10:00',
  '11:00',
```

```
  existingAppointments
);
```

📈 Estadísticas y Métricas

El sistema calcula automáticamente:

```
interface ScheduleStats {
  totalWorkingHours: 45,      // Horas semanales
  workingDays: 5,              // Días laborables
  averageHoursPerDay: 9,       // Promedio por día
  totalExceptions: 5,          // Total de excepciones
  upcomingExceptions: 3        // Excepciones futuras
}
```

🔄 Integración con Sistema de Citas

Validación al Crear Cita

```
// En appointmentService.ts
const professional = await getProfessional(professionalId);
const scheduleConfig = professional.scheduleConfig as
        ScheduleConfig;

const isAvailable = ScheduleManager.isAvailable(
  scheduleConfig,
  appointmentDate,
  startTime,
  endTime,
  existingAppointments
);

if (!isAvailable) {
  throw new Error('El profesional no está disponible en ese
        horario');
}
```

Mostrar Slots Disponibles

```
// En formulario de citas
const availableSlots = ScheduleManager.calculateAvailableSlots(
  scheduleConfig,
  {
    professionalId,
    startDate: selectedDate,
    endDate: selectedDate,
    serviceDuration: service.duration
```

```
  },
  existingAppointments
);

// Mostrar slots en UI
availableSlots.map(slot => (
  <button>{slot.startTime} - {slot.endTime}</button>
));
```

🔦 Testing

Casos de Prueba Recomendados

1. **Validación de Horarios**
   - ✅ Formato de hora válido/inválido
   - ✅ Hora fin > hora inicio
   - ✅ Duración mínima
   - ✅ Detección de solapamientos
2. **Gestión de Excepciones**
   - ✅ Agregar excepción
   - ✅ Eliminar excepción
   - ✅ Excepción sobrescribe horario regular
3. **Cálculo de Disponibilidad**
   - ✅ Slots en horario regular
   - ✅ Slots con excepciones
   - ✅ Slots con citas existentes
   - ✅ Respeto de duración de servicio
4. **Estadísticas**
   - ✅ Cálculo de horas semanales
   - ✅ Conteo de días laborables
   - ✅ Promedio de horas por día

🐛 Manejo de Errores

Errores Comunes y Soluciones

1. **“Hora de fin debe ser posterior a hora de inicio”**
   - Verificar formato de horas
   - Asegurar que endTime > startTime
2. **“Bloques de tiempo solapados”**
   - Revisar horarios del día
   - Eliminar o ajustar bloques conflictivos
3. **“Día laboral sin horarios definidos”**
   - Agregar al menos un bloque de tiempo
   - O desmarcar como día laboral

4. **“Profesional no disponible en ese horario”**
    - Verificar horario del profesional
    - Revisar excepciones
    - Verificar citas existentes

📝 Notas de Implementación

Decisiones de Diseño

1. **Almacenamiento en JSON**
    - Flexibilidad para cambios futuros
    - No requiere migraciones para ajustes menores
    - Fácil versionado del schema
2. **Validación en Múltiples Capas**
    - Frontend: UX inmediata
    - Backend: Seguridad y consistencia
    - Service: Lógica de negocio
3. **Timezone Support**
    - Preparado para múltiples zonas horarias
    - Default: America/Mexico_City
4. **Versioning del Schema**
    - Campo `version` para migraciones futuras
    - Permite evolución del formato

Limitaciones Conocidas

1. **Sin Soporte para Horarios Rotativos**
    - Actualmente solo horario semanal fijo
    - Futuro: Horarios por semana/mes
2. **Sin Sincronización con Calendarios Externos**
    - Futuro: Integración con Google Calendar, iCal
3. **Sin Notificaciones de Cambios**
    - Futuro: Notificar a clientes cuando cambia horario

🔮 Próximas Mejoras (Fase 2-5)

Fase 2: Asignación Masiva

- Asignar profesionales a múltiples sucursales
- Horarios específicos por sucursal
- Importación/exportación de asignaciones

Fase 3: Reportes Avanzados

- Reportes de productividad por profesional

- Análisis de ocupación
- Comparativas entre profesionales

Fase 4: Vista de Calendario

- Calendario visual por profesional
- Drag & drop de citas
- Vista semanal/mensual

Fase 5: Notificaciones Avanzadas

- Notificaciones de cumpleaños
- Recordatorios automáticos
- Alertas de cambios de horario

🖨 Recursos Adicionales

Documentación Relacionada

- CHECKPOINT_v1.4.0.md - Estado previo
- README.md - Documentación general
- TECHNICAL_GUIDE.md - Guía técnica

Archivos Modificados/Creados

```
app/
├── lib/
│   ├── types/
│   │   └── schedule.ts                    [NUEVO]
│   └── services/
│       └── scheduleManager.ts             [NUEVO]
├── api/
│   └── professionals/
│       └── [id]/
│           └── schedule/
│               └── route.ts               [NUEVO]
├── components/
│   └── ScheduleManager.tsx                [NUEVO]
└── dashboard/
    └── professionals/
        └── schedule/
            └── [id]/
                └── page.tsx               [NUEVO]
```

✔ Checklist de Implementación

- ☒ Definir tipos y estructuras de datos
- ☒ Implementar ScheduleManager service
- ☒ Crear endpoints API
- ☒ Desarrollar componente ScheduleManager
- ☒ Crear página de gestión de horarios
- ☒ Implementar validaciones
- ☒ Agregar manejo de errores
- ☒ Documentar código
- ☒ Crear documentación de usuario
- ☒ Preparar para integración con citas

🎉 Conclusión

La Fase 1 implementa exitosamente un sistema robusto y flexible de gestión de horarios para profesionales. El sistema está listo para:

1. ✅ Configurar horarios personalizados por profesional
2. ✅ Gestionar múltiples bloques de tiempo por día
3. ✅ Manejar excepciones y días especiales
4. ✅ Validar disponibilidad al crear citas
5. ✅ Calcular slots disponibles automáticamente

El código es mantenible, escalable y está preparado para las siguientes fases del proyecto.

Implementado por: Sistema de Desarrollo CitaPlanner
Fecha: 14 de Octubre, 2025
Versión: 1.5.0
Estado: ✅ Completado y Listo para Producción