Plan de Desarrollo Frontend: Clínica Odontológica (React + Vite + PrimeReact)

Este documento describe el desarrollo completo del frontend de la aplicación de gestión de clínica odontológica Clinica3S, implementado con React, Vite y la suite de componentes PrimeReact.

🛠️ Paso 0: Stack Tecnológico e Inicialización

Stack Tecnológico Implementado:

- **Framework:** React 18+ con JavaScript
- **Build Tool:** Vite
- **UI Library:** PrimeReact 10+ con PrimeIcons
  - Tema configurado: lara-light-indigo
  - PrimeFlex para sistema de rejilla y utilidades CSS
- **Routing:** react-router-dom v6+
- **HTTP Client:** axios
- **Gráficas:** Componente Chart de PrimeReact con chart.js
- **Gestión de Estado:** Context API (AuthContext)
- **Formateo:** date-fns y formatters personalizados

Estructura del Proyecto:

```
src/
├── api/          # Configuración de axios
├── components/   # Componentes reutilizables
│   ├── appointments/
│   ├── common/
│   ├── dashboard/
│   └── layout/
├── context/      # Context API
├── pages/        # Vistas principales
├── services/     # Servicios API
└── utils/        # Utilidades y constantes
```

🔐 Paso 1: Autenticación y Seguridad

Implementación realizada:

1. Axios Instance (api/api.js):

- BaseURL configurada: `http://localhost:8080`
- Interceptor de peticiones: Inyecta JWT Bearer token automáticamente
- Interceptor de respuestas: Maneja errores 401 (redirige a login) y 403 (sin permisos)
- Gestión de tokens en localStorage

2. AuthContext (context/AuthContext.jsx):

- Estado global con Context API de React
- Datos almacenados: `user`, `token`, `role`, `userId`, `dentistId` (para dentistas)
- Funciones implementadas:
  - `login(username, password)`: Autentica y almacena JWT
  - `logout()`: Limpia sesión y redirige a login
  - `hasRole(roles)`: Verifica permisos por rol
- Persistencia de sesión usando localStorage
- Notificaciones con `<Toast>` de PrimeReact

3. Rutas Protegidas (components/common/ProtectedRoute.jsx):

- Componente wrapper para react-router-dom
- Verifica autenticación antes de renderizar rutas
- Redirige a `/login` si no está autenticado

4. Servicio de Autenticación (services/authService.js):

- `login(credentials)`: POST `/api/auth/login`
- `changePassword(data)`: PUT `/api/auth/change-password`
- Respuesta del login incluye: `token`, `role`, `username`, `userId`, `dentistId`

Roles del Sistema:

- **ADMIN**: Acceso completo a todas las funcionalidades
- **RECEPTIONIST**: Gestión de citas, pacientes (sin acceso a usuarios)
- **DENTIST**: Consulta de sus propias citas y agenda

🎨 Paso 2: Layout y Navegación

Implementación realizada:

Layout Principal (components/layout/MainLayout.jsx):

- Usa `<Menubar>` de PrimeReact para navegación superior
- Menú adaptado dinámicamente según el rol del usuario:
  - **ADMIN**: Todas las opciones (Dashboard, Citas, Pacientes, Dentistas, Servicios, Especialidades, Usuarios, Perfil)
  - **RECEPTIONIST**: Dashboard, Citas, Pacientes, Servicios, Especialidades, Perfil
  - **DENTIST**: Solo agenda de sus citas y perfil
- Logo y nombre de la clínica en la izquierda
- Botón de logout con icono y confirmación
- Avatar con iniciales del usuario

Página Login (pages/LoginPage.jsx):

- Diseño centrado con `<Card>` de PrimeReact
- Componentes utilizados:
  - `<InputText>` para username
  - `<Password>` para contraseña (sin feedback visual)

    - `<Button>` para submit con loading state
    - `<Toast>` para notificaciones de error/éxito
- Validación de campos requeridos
- Logo de la clínica en el header

Página de Perfil (pages/ProfilePage.jsx):

- Cambio de contraseña con validación
- Campos: contraseña actual, nueva contraseña, confirmar contraseña
- Validación de longitud mínima (6 caracteres)
- Verificación de coincidencia de contraseñas

📊 Paso 3: Dashboard de Business Intelligence

Implementación realizada:

Vista Principal (pages/DashboardPage.jsx):

Selector de Año:

- `<Dropdown>` para filtrar estadísticas por año
- Rango: 2025 hasta año siguiente al actual
- Opción "Todos los años" para vista completa
- Recarga automática de datos al cambiar el año

KPIs Implementados (8 tarjetas - components/dashboard/KpiCard.jsx):

Primera Fila:

1. **Total Pacientes**: Número total de pacientes registrados
2. **Total Citas**: Con subtítulo mostrando citas completadas
3. **Citas Pendientes**: Citas futuras agendadas
4. **Sin Cobrar**: Citas atendidas pero no pagadas (unpaidAppointments)

Segunda Fila: 5. **Facturación Total**: Total cobrado a clientes (totalInvoicing) 6. **Coste Total**: Coste de tratamientos realizados (totalCost) 7. **Ganancia Neta**: Ingresos menos costes (totalRevenue) 8. **Pagos Pendientes**: Dinero por cobrar (pendingPayments)

Gráficas con Chart.js:

1. **Ingresos por Dentista** (Gráfico de Barras):
    - `<Chart type="bar">`
    - Dos series: Ingresos totales y Comisiones
    - Colores diferenciados (primary y success)
    - Formato de moneda en eje Y
2. **Estado de Citas** (Gráfico de Dona):
    - `<Chart type="doughnut">`
    - Distribución: Completadas, Pendientes, No asistió
    - Colores semafóricos: verde, amarillo, rojo

    - Leyenda en parte inferior

3. **Tendencia Mensual** (Gráfico de Líneas):
    - `<Chart type="line">`
    - Dos ejes Y: Ingresos (izquierda) y Citas (derecha)
    - Colores coordinados con los valores del eje Y
    - Ordenamiento cronológico de meses (YYYY-MM)
    - Área rellena bajo la línea de ingresos
    - Formato de moneda en eje Y izquierdo

Servicio API (services/reportService.js):

- `getDashboardStats(year)`: GET `/api/reports/dashboard?year=YYYY`
- Respuesta incluye: KPIs, monthlyStats, dentistStats

📅 Paso 4: Gestión de Citas

Implementación realizada:

Vista Principal - Calendario Visual (pages/HomePage.jsx):

Características del Calendario:

- Diseño de calendario semanal tipo Google Calendar
- Vista diaria con navegación entre días
- Horario de 8:00 a 20:00 (13 franjas horarias)
- Columna por cada dentista activo

Header del Calendario:

- Gradiente morado/azul atractivo
- Columna de horas con icono de reloj
- Avatar circular para cada dentista con su especialidad
- Estilo glassmorphism con backdrop-filter

Columna de Horas:

- Gradiente gris suave
- Texto en color morado (#6366f1)
- Fuente semibold con sombra sutil
- Sticky scroll para siempre visible

Celdas de Citas:

- Columnas intercaladas con colores de fondo sutiles
    - Pares: #fafbfc (más suave)
    - Impares: #f8f9fa (ligeramente menos suave)
- Citas con borde izquierdo según estado:
    - Verde: COMPLETED
    - Naranja: PENDING

    - Rojo: NO_SHOW
- Hover con sombra para destacar
- Click para ver detalles
- `<Tooltip>` con información resumida

Navegación:

- Botones anterior/siguiente día
- Botón "Hoy" para volver a fecha actual
- `<Calendar>` de PrimeReact para selección rápida
- Fecha formateada en español
- Botón "Nueva Cita" para crear

Leyenda:

- Indicadores de colores para cada estado
- Descripción clara de cada estado

Lista de Citas con Filtros (pages/AppointmentsPage.jsx):

Tabla Principal:

- `<DataTable>` paginado de PrimeReact
- Ordenamiento por columna
- 10 registros por página

Columnas:

1. Fecha y Hora (formateada)
2. Paciente (nombre)
3. Dentista (nombre)
4. Estado (`<Tag>` con colores)
5. Importe Total (formateado)
6. Acciones (ver detalles, pagar si hay deuda)

Filtros Implementados:

- Rango de fechas con `<Calendar>` (startDate, endDate)
- Estado de cita con `<Dropdown>` (PENDING, COMPLETED, NO_SHOW)
- Dentista con `<Dropdown>` (lista de dentistas activos)
- Paciente con `<Dropdown>` (lista de pacientes)
- Botón "Limpiar Filtros" para resetear

Acciones por Cita:

- Ver Detalles: `<Button icon="pi pi-search">`
- Registrar Pago: `<Button icon="pi pi-wallet">` (solo si tiene deuda y permisos)
- Cambiar Estado: Desde el dialog de detalles

Dialog de Detalles (components/appointments/AppointmentDetailDialog.jsx):

Información Mostrada:

- Fecha y hora de la cita
- Estado actual con `<Tag>` colorido
- Información del paciente (nombre, teléfono, email)
- Información del dentista (nombre, matrícula)
- Total a pagar (destacado)

Tabla de Servicios:

- Lista de servicios/tratamientos de la cita
- Columnas: Servicio, Cantidad, Precio, Estado Pago
- Indicador visual si está pagado (checkmark verde)
- Fecha de pago si aplica

Acciones Disponibles:

- Cambiar Estado: `<Dropdown>` con estados disponibles
- Registrar Pago: Botón destacado (solo ADMIN/RECEPTIONIST)
- Confirmación con `<ConfirmDialog>` antes de ejecutar

Servicio API (services/appointmentService.js):

- `getAppointments(params)`: GET `/api/appointments` con filtros
- `getAppointment(id)`: GET `/api/appointments/{id}`
- `createAppointment(data)`: POST `/api/appointments`
- `updateAppointmentStatus(id, status)`: PUT `/api/appointments/{id}/status?status=X`
- `payAppointment(id)`: PUT `/api/appointments/{id}/pay`

Estados de Cita:

- `PENDING`: Cita agendada, no realizada
- `COMPLETED`: Cita realizada (pagada o no)
- `NO_SHOW`: Paciente no asistió

📝 Paso 5: Formulario de Nueva Cita

Implementación realizada (pages/NewAppointmentPage.jsx):

Estructura del Formulario:

- Diseño en `<Card>` con título y breadcrumb
- Layout responsivo con grid de PrimeFlex
- Tres secciones principales: Cabecera, Servicios, Resumen

Sección 1: Datos de la Cita

Campos Implementados:

1. **Paciente**:
    - `<Dropdown>` con búsqueda
    - Carga lista completa de pacientes activos

    - Muestra nombre completo
    - Requerido

2. **Dentista**:
    - `<Dropdown>` con lista de dentistas
    - Muestra nombre y especialidad
    - Filtrado por especialidad si es necesario
    - Requerido

3. **Fecha y Hora**:
    - `<Calendar showTime hourFormat="24">`
    - Selector de fecha y hora combinado
    - Formato: DD/MM/YYYY HH:mm
    - Validación de fecha futura
    - Requerido

Sección 2: Servicios

Selector de Servicios:

- `<Dropdown>` para seleccionar servicio
- Lista filtrada por especialidad del dentista seleccionado
- Muestra nombre y precio del servicio
- `<InputNumber>` para cantidad con botones +/-
- Botón "Agregar Servicio" con validación

Tabla de Servicios Seleccionados:

- `<DataTable>` pequeño en línea
- Columnas:
    - Servicio (nombre)
    - Cantidad (editable con `<InputNumber>`)
    - Precio Unitario (formateado)
    - Subtotal (calculado)
    - Acción (eliminar servicio)
- Actualización en tiempo real del total
- Validación: mínimo 1 servicio requerido

Sección 3: Resumen

Cálculos Automáticos:

- Total de la cita calculado en tiempo real
- Formato de moneda con formatCurrency()
- Destacado visual del monto total

Botones de Acción:

- **Guardar**: `<Button label="Guardar Cita" icon="pi pi-save" />`

    - Valida todos los campos
    - Envía petición POST al backend
    - Notificación de éxito/error con `<Toast>`
    - Redirige a lista de citas al guardar
- **Cancelar**: `<Button label="Cancelar" outlined />`
    - Confirmación con `<ConfirmDialog>`
    - Vuelve a la página anterior

Validaciones Implementadas:

- Todos los campos de cabecera son requeridos
- Fecha debe ser válida y no en el pasado
- Al menos un servicio debe estar agregado
- Cantidad de servicios debe ser >= 1
- Mensajes de error claros con `<Toast>`

Formato del Request:

```
{
  "dateTime": "2026-01-15T10:00:00",
  "patientId": 5,
  "dentistId": 2,
  "services": [
    {
      "serviceId": 3,
      "quantity": 1
    },
    {
      "serviceId": 7,
      "quantity": 2
    }
  ]
}
```

Servicio API:

- `createAppointment(data)`: POST `/api/appointments`
- Respuesta: Objeto `AppointmentResponse` con ID generado

⚙️ Paso 6: Gestión de Pacientes

Implementación realizada (pages/PatientsPage.jsx):

Vista Principal:

Header con Acciones:

- Título "Gestión de Pacientes"
- Botón "Nuevo Paciente" (solo ADMIN/RECEPTIONIST)
- `<Toolbar>` con acciones principales

Filtros de Búsqueda:

- Nombre: `<InputText>` con búsqueda parcial
- Teléfono: `<InputText>` con búsqueda parcial
- Email: `<InputText>` con búsqueda parcial
- Botón "Buscar" y "Limpiar"
- Búsqueda en tiempo real al escribir

Tabla de Pacientes:

- `<DataTable>` paginado (10 registros por página)
- Columnas:
    1. ID
    2. Nombre
    3. Fecha de Nacimiento (formateada)
    4. Género
    5. Teléfono
    6. Email
    7. Acciones (Editar, Eliminar)
- Ordenamiento por columna
- Estados de carga con `<ProgressSpinner>`

Dialog de Edición/Creación:

Campos del Formulario:

1. **Nombre**: `<InputText>` (requerido)
2. **Fecha de Nacimiento**: `<Calendar>` con formato DD/MM/YYYY (requerido)
3. **Género**: `<Dropdown>` con opciones Masculino/Femenino/Otro (requerido)
4. **Teléfono**: `<InputText>` (requerido)
5. **Email**: `<InputText>` con validación de formato (requerido)

Validaciones:

- Todos los campos son obligatorios
- Email con formato válido
- Fecha de nacimiento en el pasado
- Mensajes de error específicos

Botones:

- Guardar: Crea o actualiza según el modo
- Cancelar: Cierra el dialog sin guardar

Confirmación de Eliminación:

- `<ConfirmDialog>` antes de eliminar
- Mensaje: "¿Estás seguro de eliminar este paciente?"
- Notificación de éxito/error

Servicio API (services/patientService.js):

- `getPatients(params)`: GET `/api/patients?name=X&phone=Y&email=Z&page=0&size=10`
- `getPatient(id)`: GET `/api/patients/{id}`
- `createPatient(data)`: POST `/api/patients`
- `updatePatient(id, data)`: PUT `/api/patients/{id}`
- `deletePatient(id)`: DELETE `/api/patients/{id}`

Modelo de Datos - Patient:

```
{
  "id": 1,
  "name": "Juan Pérez",
  "birthDate": "1985-03-15",
  "gender": "Masculino",
  "phone": "+34123456789",
  "email": "juan.perez@example.com"
}
```

⚙️ Paso 7: Gestión de Dentistas

Implementación realizada (pages/DentistsPage.jsx):

Vista Principal:

Header:

- Título "Gestión de Dentistas"
- Botón "Nuevo Dentista" (solo ADMIN)

Filtros:

- Nombre: `<InputText>` con búsqueda parcial
- Especialidad: `<Dropdown>` con lista de especialidades
- Botones "Buscar" y "Limpiar"

Tabla de Dentistas:

- `<DataTable>` paginado
- Columnas:
    1. ID
    2. Nombre (desde user.name)
    3. Matrícula Profesional
    4. Especialidades (chips con `<Tag>`)
    5. Tasa de Comisión (%)
    6. Usuario Activo (badge)
    7. Acciones (Editar, Eliminar)

Dialog de Edición/Creación:

Campos del Formulario:

1. **Nombre Completo**: `<InputText>` (requerido)
2. **Matrícula Profesional**: `<InputText>` (requerido, único)
3. **Especialidades**: `<MultiSelect>` para seleccionar múltiples (requerido)
    - Muestra checkboxes para cada especialidad
    - Permite selección múltiple
    - Lista obtenida de API
4. **Tasa de Comisión**: `<InputNumber>` con % (requerido)
    - Rango: 0-100
    - Formato: 2 decimales
    - Muestra símbolo %

Solo en Modo Creación: 5. **Nombre de Usuario**: `<InputText>` (requerido, único) 6. **Contraseña**: `<Password>` (requerido, mínimo 6 caracteres)

Confirmación de Eliminación:

- `<ConfirmDialog>` con advertencia
- Mensaje: "¿Eliminar este dentista? Esto puede afectar citas asociadas"

Servicio API (services/dentistService.js):

- `getDentists(params)`: GET `/api/dentists?name=X&specialtyId=Y&page=0&size=10`
- `getDentist(id)`: GET `/api/dentists/{id}`
- `updateDentist(id, data)`: PUT `/api/dentists/{id}`
- `deleteDentist(id)`: DELETE `/api/dentists/{id}`

Nota: La creación de dentistas se hace desde gestión de usuarios (UsersPage) porque crea tanto el usuario como el dentista asociado.

Modelo de Datos - Dentist:

```
{
  "id": 1,
  "licenseNumber": "ODO-12345",
  "commissionRate": 15.50,
  "user": {
    "id": 10,
    "username": "dr.garcia",
    "name": "Dr. Juan García",
    "role": "DENTIST",
    "enabled": true
  },
  "specialties": [
    { "id": 1, "name": "Ortodoncia" },
    { "id": 3, "name": "Endodoncia" }
  ]
}
```

⚙️ Paso 8: Gestión de Servicios

Implementación realizada (pages/ServicesPage.jsx):

Vista Principal:

Header:

- Título "Gestión de Servicios Odontológicos"
- Botón "Nuevo Servicio" (solo ADMIN)

Tabla de Servicios:

- `<DataTable>` paginado
- Columnas:
    1. ID
    2. Nombre del Servicio
    3. Especialidad (`<Tag>` colorido)
    4. Coste Estándar (formateado)
    5. Precio de Lista (formateado)
    6. Margen (calculado y mostrado en %)
    7. Acciones (Editar, Eliminar)
- Ordenamiento por columna

Dialog de Edición/Creación:

Campos del Formulario:

1. **Nombre**: `<InputText>` (requerido)
2. **Especialidad**: `<Dropdown>` con lista de especialidades (requerido)
3. **Coste Estándar**: `<InputNumber>` con formato moneda (requerido)
    - Mínimo: 0
    - 2 decimales
    - Prefijo: €
4. **Precio de Lista**: `<InputNumber>` con formato moneda (requerido)
    - Mínimo: 0
    - 2 decimales
    - Prefijo: €
    - Validación: debe ser >= Coste Estándar

Cálculo Automático:

- Margen de Ganancia mostrado en tiempo real
- Fórmula: ((Precio - Coste) / Precio) * 100

Confirmación de Eliminación:

- `<ConfirmDialog>` con advertencia
- Mensaje: "¿Eliminar este servicio? Podría afectar citas futuras"

Servicio API (services/serviceService.js):

- `getServices(params)`: GET `/api/services?page=0&size=10`

- `getService(id)`: GET `/api/services/{id}`
- `getServicesBySpecialty(specialtyId)`: GET `/api/services/specialty/{specialtyId}`
- `createService(data)`: POST `/api/services`
- `updateService(id, data)`: PUT `/api/services/{id}`
- `deleteService(id)`: DELETE `/api/services/{id}`

Modelo de Datos - Service:

```
{
  "id": 1,
  "name": "Limpieza Dental",
  "standardCost": 25.00,
  "listPrice": 50.00,
  "specialty": {
    "id": 1,
    "name": "Odontología General"
  }
}
```

⚙️ Paso 9: Gestión de Especialidades

Implementación realizada (pages/SpecialtiesPage.jsx):

Vista Principal:

Header:

- Título "Especialidades Odontológicas"
- Botón "Nueva Especialidad" (solo ADMIN)

Tabla de Especialidades:

- `<DataTable>` simple sin paginación (pocas especialidades)
- Columnas:
    1. ID
    2. Nombre
    3. Acciones (Editar, Eliminar)
- Ordenamiento alfabético

Dialog de Edición/Creación:

Campos del Formulario:

1. **Nombre**: `<InputText>` (requerido, único)
    - Ej: "Ortodoncia", "Endodoncia", "Periodoncia"

Confirmación de Eliminación:

- `<ConfirmDialog>` con advertencia fuerte

- Mensaje: "¿Eliminar esta especialidad? Afectará a dentistas y servicios"

Servicio API (services/specialtyService.js):

- `getSpecialties()`: GET `/api/specialties`
- `getSpecialty(id)`: GET `/api/specialties/{id}`
- `createSpecialty(data)`: POST `/api/specialties`
- `updateSpecialty(id, data)`: PUT `/api/specialties/{id}`
- `deleteSpecialty(id)`: DELETE `/api/specialties/{id}`

Modelo de Datos - Specialty:

```
{
  "id": 1,
  "name": "Ortodoncia"
}
```

Especialidades Comunes:

- Odontología General
- Ortodoncia
- Endodoncia
- Periodoncia
- Cirugía Oral
- Implantología
- Estética Dental
- Odontopediatría

⚙️ Paso 10: Gestión de Usuarios (Solo ADMIN)

Implementación realizada (pages/UsersPage.jsx):

Vista Principal:

Header:

- Título "Gestión de Usuarios del Sistema"
- Botón "Nuevo Usuario" (solo ADMIN)

Tabla de Usuarios:

- `<DataTable>` con todos los usuarios
- Columnas:
    1. ID
    2. Nombre de Usuario (username)
    3. Nombre Completo
    4. Rol (`<Tag>` colorido: ADMIN-rojo, DENTIST-azul, RECEPTIONIST-verde)
    5. Estado (Activo/Inactivo con `<Tag>`)
    6. Acciones (Habilitar/Deshabilitar, Eliminar)

- Sin filtros (lista completa)

Dialog de Creación de Usuario:

Campos del Formulario:

1. **Nombre de Usuario**: `<InputText>` (requerido, único)
2. **Contraseña**: `<Password>` (requerido, mínimo 6 caracteres)
3. **Rol**: `<Dropdown>` con opciones (requerido)
    - ADMIN
    - DENTIST
    - RECEPTIONIST

Si Rol = DENTIST (campos adicionales): 4. **Nombre Completo del Dentista**: `<InputText>` (requerido) 5. **Matrícula Profesional**: `<InputText>` (requerido, único) 6. **Tasa de Comisión**: `<InputNumber>` % (requerido, 0-100) 7. **Especialidades**: `<MultiSelect>` (requerido, mínimo 1)

Acciones Disponibles:

- **Crear Usuario**: Crea user y dentista si aplica
- **Habilitar/Deshabilitar**: Toggle del estado enabled
- **Eliminar**: Solo si no tiene datos asociados

Confirmaciones:

- Cambio de estado: "¿Cambiar estado del usuario X?"
- Eliminación: "¿Eliminar usuario X? Esta acción no se puede deshacer"

Servicio API (services/userService.js):

- `getUsers()`: GET `/api/users`
- `getUser(id)`: GET `/api/users/{id}`
- `createUser(data)`: POST `/api/users`
- `toggleUserEnabled(id, enabled)`: PUT `/api/users/{id}/enabled?enabled=true/false`
- `deleteUser(id)`: DELETE `/api/users/{id}`

Modelo de Datos - AppUser:

```
{
  "id": 1,
  "username": "admin",
  "name": "Administrador",
  "role": "ADMIN",
  "enabled": true
}
```

Request de Creación (RegisterRequest):

```
{
  "username": "dr.martinez",
  "password": "password123",
  "role": "DENTIST",
  "dentistName": "Dr. Carlos Martínez",
  "licenseNumber": "ODO-67890",
  "commissionRate": 18.00,
  "specialtyIds": [1, 3, 5]
}
```

🛠️ Paso 11: Utilidades y Constantes

Implementación realizada:

Formatters (utils/formatters.js):

Funciones de Formateo:

1. **formatCurrency(value)**: Formatea números como moneda
    - Símbolo: €
    - Decimales: 2
    - Separador de miles: .
    - Separador decimal: ,
    - Ejemplo: 1500.50 → "1.500,50 €"
2. **formatNumber(value)**: Formatea números enteros
    - Separador de miles: .
    - Ejemplo: 1500 → "1.500"
3. **formatDate(date)**: Formatea fechas
    - Formato: DD/MM/YYYY
    - Ejemplo: "2026-01-15" → "15/01/2026"
4. **formatDateTime(date)**: Formatea fecha y hora
    - Formato: DD/MM/YYYY HH:mm
    - Ejemplo: "2026-01-15T14:30:00" → "15/01/2026 14:30"

Constantes (utils/constants.js):

STATUS_COLORS - Estados de Citas:

```
{
  PENDING: {
    label: 'Pendiente',
    color: '#f59e0b',
    background: '#fef3c7'
  },
```

```
  COMPLETED: {
    label: 'Completada',
    color: '#10b981',
    background: '#d1fae5'
  },
  NO_SHOW: {
    label: 'No asistió',
    color: '#ef4444',
    background: '#fee2e2'
  }
}
```

CHART_COLORS - Colores para Gráficas:

```
{
  primary: '#6366f1',
  success: '#10b981',
  warning: '#f59e0b',
  danger: '#ef4444',
  info: '#3b82f6',
  secondary: '#6b7280'
}
```

ROLES - Roles del Sistema:

```
{
  ADMIN: 'ADMIN',
  DENTIST: 'DENTIST',
  RECEPTIONIST: 'RECEPTIONIST'
}
```

Componentes Reutilizables:

KpiCard (components/dashboard/KpiCard.jsx):

- Props: title, value, icon, color, subtitle
- Muestra tarjeta con métrica
- Iconos de PrimeIcons
- Colores: primary, success, warning, danger, info, secondary

ProtectedRoute (components/common/ProtectedRoute.jsx):

- Wrapper para rutas protegidas
- Verifica autenticación
- Redirige a /login si no autenticado

AppointmentDetailDialog (components/appointments/AppointmentDetailDialog.jsx):

- Dialog reutilizable para detalles de cita
- Muestra información completa
- Acciones: pagar, cambiar estado
- Usado en HomePage y AppointmentsPage

📋 Paso 12: Modelo de Datos y API

Modelo de Datos Principal:

Cita (AppointmentResponse):

```
{
  "id": 1,
  "dateTime": "2026-01-15T10:00:00",
  "status": "PENDING",
  "totalAmount": 150.00,
  "patient": {
    "id": 5,
    "name": "María González",
    "phone": "+34666777888",
    "email": "maria@example.com"
  },
  "dentist": {
    "id": 2,
    "name": "Dr. Juan García",
    "licenseNumber": "ODO-12345"
  },
  "details": [
    {
      "id": 1,
      "serviceName": "Limpieza Dental",
      "quantity": 1,
      "priceApplied": 50.00,
      "paid": true,
      "paymentDate": "2026-01-15T11:00:00"
    },
    {
      "id": 2,
      "serviceName": "Blanqueamiento",
      "quantity": 1,
      "priceApplied": 100.00,
      "paid": false,
      "paymentDate": null
    }
  ]
}
```

Dashboard Stats (DashboardStatsDTO):

```
{
  "totalPatients": 150,
  "totalAppointments": 450,
  "completedAppointments": 380,
  "unpaidAppointments": 25,
  "pendingAppointments": 70,
  "totalRevenue": 45000.00,
  "totalInvoicing": 65000.00,
  "totalCost": 20000.00,
  "pendingPayments": 5000.00,
  "monthlyStats": [
    {
      "month": "2026-01",
      "appointments": 45,
      "revenue": 4500.00
    }
  ],
  "dentistStats": [
    {
      "dentistId": 1,
      "dentistName": "Dr. García",
      "appointments": 120,
      "revenue": 12000.00,
      "commission": 1800.00
    }
  ]
}
```

Endpoints de la API:

Autenticación:

- POST `/api/auth/login` - Iniciar sesión
- PUT `/api/auth/change-password` - Cambiar contraseña

Citas:

- GET `/api/appointments` - Listar con filtros (paginado)
- GET `/api/appointments/{id}` - Obtener detalle
- POST `/api/appointments` - Crear nueva cita
- PUT `/api/appointments/{id}/status?status=X` - Cambiar estado
- PUT `/api/appointments/{id}/pay` - Registrar pago

Pacientes:

- GET `/api/patients` - Listar con filtros (paginado)
- GET `/api/patients/{id}` - Obtener paciente
- POST `/api/patients` - Crear paciente
- PUT `/api/patients/{id}` - Actualizar paciente
- DELETE `/api/patients/{id}` - Eliminar paciente

Dentistas:

- GET `/api/dentists` - Listar con filtros (paginado)
- GET `/api/dentists/{id}` - Obtener dentista
- PUT `/api/dentists/{id}` - Actualizar dentista
- DELETE `/api/dentists/{id}` - Eliminar dentista

Servicios:

- GET `/api/services` - Listar (paginado)
- GET `/api/services/{id}` - Obtener servicio
- GET `/api/services/specialty/{specialtyId}` - Por especialidad
- POST `/api/services` - Crear servicio
- PUT `/api/services/{id}` - Actualizar servicio
- DELETE `/api/services/{id}` - Eliminar servicio

Especialidades:

- GET `/api/specialties` - Listar todas
- GET `/api/specialties/{id}` - Obtener especialidad
- POST `/api/specialties` - Crear especialidad
- PUT `/api/specialties/{id}` - Actualizar especialidad
- DELETE `/api/specialties/{id}` - Eliminar especialidad

Usuarios:

- GET `/api/users` - Listar todos (solo ADMIN)
- GET `/api/users/{id}` - Obtener usuario
- POST `/api/users` - Crear usuario (crea dentista si aplica)
- PUT `/api/users/{id}/enabled?enabled=X` - Habilitar/deshabilitar
- DELETE `/api/users/{id}` - Eliminar usuario

Reportes:

- GET `/api/reports/dashboard?year=YYYY` - Estadísticas del dashboard

🎯 Resumen de Funcionalidades Implementadas

Gestión Completa: ✅ Sistema de autenticación con JWT ✅ Control de acceso basado en roles (RBAC) ✅ Dashboard con Business Intelligence y filtro por año ✅ Calendario visual de citas estilo Google Calendar ✅ Gestión completa de citas (crear, ver, pagar, cambiar estado) ✅ CRUD de pacientes con filtros de búsqueda ✅ CRUD de dentistas con especialidades múltiples ✅ CRUD de servicios odontológicos ✅ CRUD de especialidades ✅ Gestión de usuarios del sistema ✅ Cambio de contraseña de usuario propio ✅ Notificaciones con Toast ✅ Confirmaciones con ConfirmDialog ✅ Diseño responsivo y moderno ✅ Componentes reutilizables ✅ Manejo de errores centralizado ✅ Formateo de moneda y fechas ✅ Gráficas interactivas con Chart.js

Roles y Permisos:

Funcionalidad	ADMIN	RECEPTIONIST	DENTIST

Funcionalidad	ADMIN	RECEPTIONIST	DENTIST
Ver Dashboard	✅	✅	✅
Ver Calendario	✅	✅	✅ (solo sus citas)
Crear Citas	✅	✅	❌
Pagar Citas	✅	✅	❌
Gestionar Pacientes	✅	✅	❌
Gestionar Dentistas	✅	❌	❌
Gestionar Servicios	✅	Ver	Ver
Gestionar Especialidades	✅	Ver	Ver
Gestionar Usuarios	✅	❌	❌
Cambiar Contraseña Propia	✅	✅	✅

Tecnologías y Librerías:

- React 18 con Hooks
- Vite (build tool)
- React Router DOM v6
- PrimeReact 10 (componentes UI)
- PrimeIcons (iconos)
- PrimeFlex (CSS utilities)
- Axios (HTTP client)
- Chart.js con PrimeReact Chart
- date-fns (manipulación de fechas)
- Context API (gestión de estado)

Patrones y Buenas Prácticas:

- Arquitectura basada en componentes
- Separación de concerns (services, components, pages)
- Context API para estado global
- Custom Hooks para lógica reutilizable
- Interceptores de Axios para manejo centralizado
- Manejo de errores consistente
- Validación de formularios
- Confirmaciones para acciones críticas
- Diseño responsivo mobile-first
- Código modular y reutilizable