Memoria del Proyecto: Clinica3S

Asignatura: Sistemas de Información | **Titulación:** MEI | **Fecha:** Enero 2026 | **Autor:** Edgar Conde

1. Descripción del Contexto

Clinica3S es un sistema de gestión integral para clínicas dentales que digitaliza la gestión de citas, pacientes, servicios y análisis financiero mediante Business Intelligence.

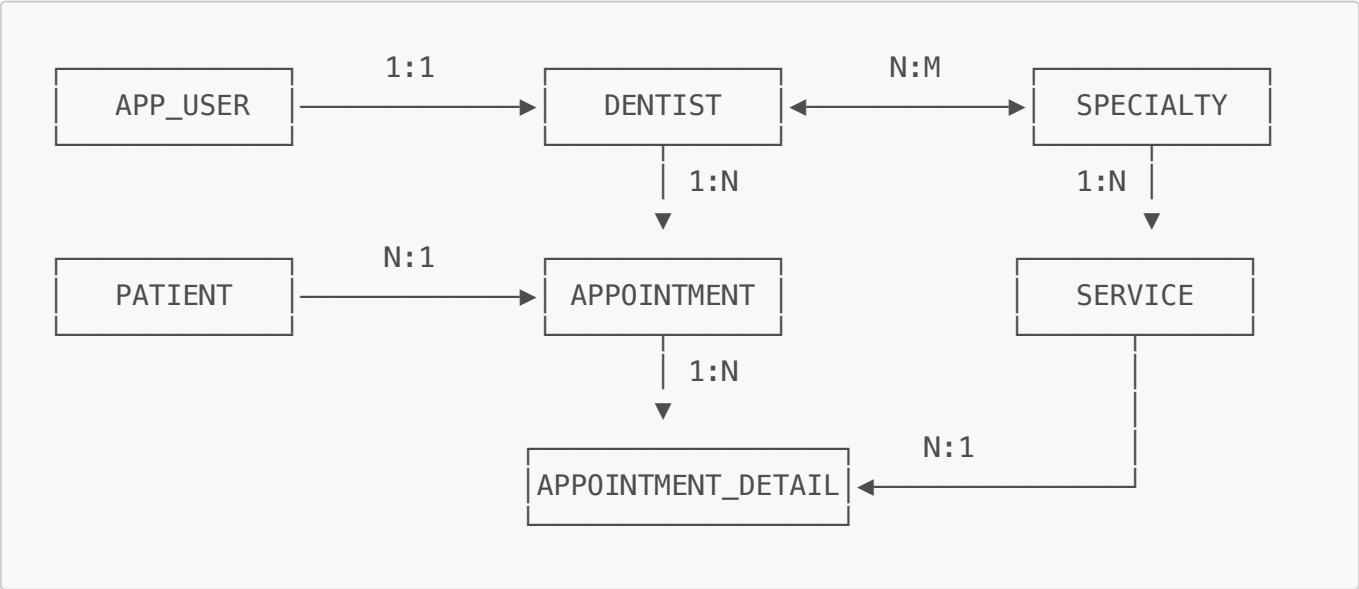
Problemática y Solución

Problema	Solución
Agendas manuales desconectadas	Calendario visual con estados de seguimiento
Falta de visibilidad financiera	Dashboard BI con KPIs e informes
Historiales dispersos	Administración centralizada de pacientes
Asignación de recursos compleja	Gestión de dentistas por especialidades

Roles del Sistema

Rol	Permisos
ADMIN	Acceso completo: usuarios, configuración, reportes
RECEPTIONIST	Citas, pacientes, servicios (sin usuarios ni reportes)
DENTIST	Consulta de agenda y citas propias

2. Modelo de Datos



Entidades

Entidad	Campos Clave
APP_USER	username, password (BCrypt), role (ADMIN/DENTIST/RECEPTIONIST), enabled
DENTIST	licenseNumber, commissionRate, user_id (FK 1:1)
SPECIALTY	name (Ortodoncia, Endodoncia, Cirugía Oral, etc.)
SERVICE	name, standardCost, listPrice, specialty_id
PATIENT	name, birthDate, gender, phone, email
APPOINTMENT	dateTime, status (PENDING/COMPLETED/NO_SHOW), totalAmount, patient_id, dentist_id
APPOINTMENT_DETAIL	quantity, priceApplied, paymentDate, appointment_id, service_id

Decisiones de Diseño

- `priceApplied`: Preserva precio histórico para análisis BI
- `paymentDate`: Control granular de pagos parciales por servicio
- `commissionRate`: Cálculo automático de comisiones

---

3. Historias de Usuario

Autenticación

- HU-01/02: Login y cambio de contraseña
- HU-03/04: Crear y habilitar/deshabilitar usuarios (ADMIN)

Citas

- HU-05: Ver calendario de citas del día
- HU-06: Crear cita (paciente + dentista + servicios)
- HU-07/08: Listado filtrable y detalle de citas
- HU-09/10: Cambiar estado y registrar pagos
- HU-11: Agenda personal del dentista

Pacientes y Dentistas

- HU-12/15: CRUD de pacientes con búsqueda
- HU-16/19: CRUD de dentistas con especialidades

Servicios y Reportes

- HU-20/23: Gestión de especialidades y servicios con márgenes
- HU-24/29: Dashboard BI (KPIs, gráficas, tendencias, pagos pendientes)

---

4. Arquitectura y Tecnologías

Arquitectura

- **Backend**: API REST Spring Boot (capas: Controller → Service → Repository)
- **Frontend**: SPA React
- **BD**: H2 (desarrollo) / PostgreSQL (producción)

Backend - Estructura

```
io.github.edconde.clinica3s_backend/
├── controller/    # REST endpoints
├── service/       # Lógica de negocio
├── repository/    # JPA
├── entity/        # Dominio
├── dto/           # Transfer Objects
├── security/      # JWT
└── exception/     # Manejo errores
```

Endpoints principales: `/api/auth`, `/api/users`, `/api/dentists`, `/api/patients`, `/api/appointments`, `/api/services`, `/api/specialties`, `/api/reports/dashboard`

Stack: Spring Boot 3.5, Spring Security + JWT, Spring Data JPA, SpringDoc OpenAPI, Lombok

Frontend - Estructura

```
src/
├── api/           # Axios config
├── context/       # AuthContext
├── components/    # Layout, ProtectedRoute, KpiCard, etc.
├── pages/         # Login, Home, Dashboard, Appointments, Patients, etc.
├── services/      # Llamadas API
└── utils/         # Constantes, formatters
```

Stack: React 19, Vite 7, PrimeReact 10, React Router 7, Chart.js, TailwindCSS

---

5. Instrucciones de Uso

Instalación

```
# Backend
cd clinica3s-backend && ./mvnw spring-boot:run
# → http://localhost:8080

# Frontend
cd clinica3s-frontend && npm install && npm run dev
# → http://localhost:5173
```

Credenciales: admin / admin123

URLs útiles: Swagger UI (`/swagger-ui.html`), H2 Console (`/h2-console`)

Variables de Entorno (Producción)

`ENVIRONMENT`, `DATABASE_URL`, `DATABASE_USERNAME`, `DATABASE_PASSWORD`, `JWT_SECRET`, `ADMIN_PASSWORD`, `CORS_ALLOWED_ORIGINS`

---

6. Conclusiones y Mejoras

Logros

- Arquitectura robusta con API REST bien definida
- Seguridad JWT con RBAC granular
- Dashboard BI para toma de decisiones
- Código mantenible con separación de responsabilidades

Problemas Resueltos

Problema	Solución
Citas con múltiples servicios	`AppointmentDetail` con precio histórico y pagos individuales
Rendimiento en consultas	Paginación y filtros en backend
Sincronización de precios	Campo `priceApplied` preserva valor histórico

Mejoras Futuras

- **Funcionales:** Notificaciones SMS/email, historial clínico, facturación PDF, multi-clínica
- **Técnicas:** Testing ampliado, caché Redis, CI/CD, PWA
- **UX:** Modo oscuro, accesibilidad WCAG, i18n

Lecciones Aprendidas

1. Almacenar datos históricos para análisis posterior
2. Implementar seguridad desde el inicio
3. Bibliotecas UI aceleran el desarrollo
4. Documentación automática (Swagger) facilita integración
5. Datos de prueba realistas son esenciales

Versión: 1.0 | Enero 2026