SCApp – Documento Técnico (Arquitectura y Tecnologías)

Este documento resume la arquitectura, tecnologías, bases de datos, y decisiones técnicas de **SCApp** para sus dos superficies: **Web** y **Móvil (Android)**.

---

1) Resumen ejecutivo

- **Front-end Web**: SPA con **Ionic + React** (Chart.js para dashboards).
- **Móvil (Android)**: **Ionic React** empaquetado con **Capacitor** → APK → JAVA.
- **Persistencia móvil**: **PouchDB** local con **sincronización** hacia **CouchDB** remoto.
- **Back-end Web**: **Spring Boot** (Java) conectado a **PostgreSQL (Neon)**.
- **Facturación**: módulo de cambio de estado a `FACTURADO` operativo; **descarga de factura** en Spring Boot **pendiente** (endpoints/estructura listos).

---

2) Tecnologías y versiones (referenciales)

Capa	Tecnología	Uso principal
Web UI	**Ionic + React**	SPA, UI responsiva, componentes móviles.
Gráficas	**Chart.js**	Línea y dona para KPIs de citas/estados.
Móvil	**Ionic React + Capacitor**	App Android (APK), acceso a capacidades nativas.
Almacenamiento local	**PouchDB**	Persistencia local, modo offline, colas de sync.
Almacenamiento remoto móvil	**CouchDB**	Base remota para replicación bidireccional.
Back-end Web	**Spring Boot (Java)**	API y servicios de negocio.
BD Web	**PostgreSQL (Neon)**	Persistencia de datos web (usuarios, facturas, etc.).
Hosting PG	**Neon**	Postgres administrado, escalable, rama/branching.
Build/Empaquetado	**Capacitor + Android SDK**	Generación de APK.

Capa	Tecnología	Uso principal
Auth	JWT / Session	Autenticación/autorización.

Las versiones exactas se definen en package.json/pom.xml y build.gradle del proyecto.

---

3) Modelado de datos (entidades principales)

Las entidades pueden existir en **CouchDB** (movil) y/o **PostgreSQL** (web).

- **cita**
  - Campos: `_id`, `type='cita'`, `clienteId`, `profesionalId`, `servicioId`, `fechaHora` (ISO), `estado` (`PROGRAMADA|CONFIRMADA|ATENDIDA|CANCELADA|NO_ASISTIO|FACTURADO`), `notas`, `createdAt`, `updatedAt`.
- **cliente**
  - Campos: `_id`, `type='cliente'`, `nombres`, `apellidos`, `clienteNac/nacimiento` (ISO), `contacto`, `createdAt`, `updatedAt`.
- **servicio**
  - Campos: `_id`, `type='servicio'`, `nombre`, `codigo`, `precio`, `activo`, `createdAt`, `updatedAt`.
- **profesional**
  - Campos: `_id`, `type='profesional'`, `nombre`, `especialidad`, `colegiado?`, `activo`, `createdAt`, `updatedAt`.
- **factura**
  - Campos: `_id`, `type='factura'`, `citaId`, `monto`, `estado` (`FACTURADO`), `emision` (ISO), `createdAt`, `updatedAt`.

---

4) Sincronización (móvil)

4.1 Estrategia

- **PouchDB ⇄ CouchDB** con **replicación continua** (live) y/o por lotes (on-demand), según vista.
- **Bidireccional**: cambios locales se suben; cambios remotos se bajan.
- **Normalización de fechas** a ISO (UTC) antes de `put`, para consistencia de IDs y ordenamiento.
- **Conflictos**: se reintenta en caso de `409` (_rev conflict); cuando procede, se hace `get → merge → put`.

4.2 Filtros y alcance

- En cargas iniciales se pueden usar **selectores** (ej.: `type='cita'`, `servicioId='quimio'`) para reducir tráfico.
- En listados se filtra por rango de fechas en el cliente para KPIs y gráficos.

4.3 Offline-first

- La app móvil funciona **offline**; los cambios quedan en cola y se sincronizan al recuperar conectividad.
- Estrategias de resiliencia: reintentos exponenciales, `updatedAt` para "last write wins" (según negocio).

---

5) Back-end Web (Spring Boot)

5.1 Estado actual

- Endpoints CRUD para módulos base (según despliegue del entorno).
- **Descarga de factura**:
  - **Pendiente de implementación** (generación PDF/archivo).
  - **Estructura y endpoints** previstos (controlador/servicio/plantilla) listos para activarse.

5.2 Integración con PostgreSQL (Neon)

- **Neon** como host de **PostgreSQL** (DBaaS administrado).
- Se utiliza **JPA/Hibernate** o JDBC Template (según módulo) y un pool de conexiones (HikariCP por defecto).

---

6) Construcción y despliegue

6.1 Web

- `npm run build` (Ionic React) → artefacto estático.
- Despliegue en hosting web (NGINX/Apache/Vercel/etc.).
- Back-end Spring Boot: empaquetado **JAR/WAR**; deploy en servidor/servicio (Railway/Render/EC2/K8s, etc.).

6.2 Móvil (Android)

- `ionic build` → `npx cap copy android` → abrir en Android Studio → generar **APK/AAB**.
- Configurar `android:usesCleartextTraffic=false` y `androidScheme='https'` en Capacitor.

- Firmado del APK (keystore).

Anexo A – Glosario breve - **PouchDB**: BD JS que corre en el cliente; replica con CouchDB. - **CouchDB**: BD NoSQL orientada a documentos con `_rev` y replicación. - **Capacitor**: puente nativo para apps híbridas con web tech. - **Neon**: Postgres administrado en la nube con escalado y branching. - **Spring Boot**: framework Java para APIs/servicios.
