

# Diccionario de Endpoints del Dashboard

---

Este archivo describe los endpoints que consume el componente `dashboard-general` y la estructura JSON que el frontend espera recibir (nombres de campos y tipos de dato). Todos los endpoints son llamados desde `ReportesService` con base en `environment.apiUrl + '/reportes'`.

`/reportes/top-enfermedades/`:

GET

```
[
  {
    "enfermedad": "Influenza",
    "casos": 125,
    "sede": "Sede Central"
  }
]
```

- Campos y tipos:
  - `enfermedad`: string
  - `casos`: number (entero)
  - `sede`: string

`/reportes/medicamentos-recetados/`:

GET

```
[
  {
    "medicamento": "Paracetamol",
    "cantidad": 240,
    "mes": "2025-11"
  }
]
```

- Campos y tipos:
  - `medicamento`: string
  - `cantidad`: number (entero)
  - `mes`: string (formato libre; ejemplo `YYYY-MM` o nombre del mes)

`/reportes/medicos-top-consultas/`:

GET

```
[
  {
    "medico": "Dra. María Pérez",
    "consultas": 48,
    "semana": "2025-W45"
  }
]
```

- Campos y tipos:
  - `medico`: string
  - `consultas`: number (entero)
  - `semana`: string (identificador de período; ejemplo YYYY-Www o rango)

/reportes/tiempos-atencion/:

GET

```
[
  {
    "sede": "Sede Norte",
    "tiempo_promedio": 15.5
  }
]
```

- Campos y tipos:
  - `sede`: string
  - `tiempo_promedio`: number (tiempo promedio en minutos). Nota: puede también ser una cadena si la API devuelve formato HH:MM:SS; en ese caso el frontend mostrará el valor tal cual.

/reportes/pacientes-por-sede/:

GET

```
[
  {
    "sede": "Sede Central",
    "total_pacientes": 1024
  }
]
```

- Campos y tipos:
  - `sede`: string
  - `total_pacientes`: number (entero)

Notas generales:

- Todos los endpoints se consumen con `GET` y retornan arreglos (`[]`) de objetos.

- Los nombres de campo mostrados arriba están basados en las plantillas del componente `dashboard-general (.html)`.
- Si la API devuelve formatos distintos (por ejemplo `tiempo_promedio` como string), el frontend simplemente mostrará el valor; si prefieres un tipo concreto (por ejemplo siempre number en minutos), conviene coordinar y normalizarlo en el backend.

---

Módulo `sedes` (archivos: `src/app/modules/sedes/services/*.ts`, `sedes-list`, `sede-form`, `sedes-detail`):

`/sedes/`:

GET

```
[
  {
    "id_sede": 1,
    "nom_sede": "Sede Central",
    "ciudad": "Ciudad X",
    "direccion": "Av. Siempre Viva 123",
    "region_id": 5,
    "region": {
      "region_id": 5,
      "nombre": "Región Norte"
    }
  }
]
```

- Campos y tipos:
  - `id_sede`: number
  - `nom_sede`: string
  - `ciudad`: string
  - `direccion`: string
  - `region_id`: number
  - `region`: object opcional con `region_id` (number) y `nombre` (string)

`/sedes/{id}/`:

GET

```
{
  "id_sede": 1,
  "nom_sede": "Sede Central",
  "ciudad": "Ciudad X",
  "direccion": "Av. Siempre Viva 123",
  "region_id": 5,
  "region": {
    "region_id": 5,
    "nombre": "Región Norte"
  }
}
```

```
}  
}
```

/sedes/:

POST

Payload esperado (crear):

```
{  
  "nom_sede": "Sede Nueva",  
  "ciudad": "Ciudad Y",  
  "direccion": "Calle Falsa 456",  
  "region_id": 3  
}
```

/sedes/{id}/:

PUT

Payload esperado (actualizar): igual que POST, incluyendo `id_sede` en la URL.

/sedes/{id}/:

DELETE

Sin payload.

Telefonos de sede (servicio `TelefonoSedeService`):

/sedes/{id}/telefonos/:

GET

```
[  
  { "id_sede": 1, "numero": "+57123456789" }  
]
```

POST /sedes/{id}/telefonos/ espera:

```
{ "id_sede": 1, "numero": "+57123456789" }
```

DELETE /sedes/{id}/telefonos/?numero=... elimina por número (query param `numero`).

/regiones/ (servicio `RegionService`):

GET

```
[
  { "region_id": 1, "nombre": "Región Centro" }
]
```

---

Módulo `equipamiento` (archivo:

`src/app/modules/equipamiento/services/equipamiento.service.ts`):

`/equipamiento/`:

GET

```
[
  {
    "cod_eq": 101,
    "nom_eq": "Monitor ECG",
    "id_dept": 4,
    "estado": "operativo",
    "fecha_mantenimiento": "2025-10-15T00:00:00Z",
    "responsable": 23,
    "responsable_empleado": { "id_emp": 23, "persona": { "nombre": "Juan Perez" } }
  }
]
```

- Campos y tipos:
  - `cod_eq`: number
  - `nom_eq`: string
  - `id_dept`: number
  - `estado`: string
  - `fecha_mantenimiento`: string|Date (ISO datetime)
  - `responsable`: number (id de empleado)
  - `responsable_empleado`: object opcional con datos del empleado/responsable

---

Módulo `personas - empleados` (archivos:

`src/app/modules/personas/services/empleado.service.ts`, `empleados-list`, `empleado-form`):

`/empleados/`:

GET

```
[
  {
    "id_emp": 23,
    "documento": "12345678",
    "id_dept": 4,
```

```

    "cargo_id": 2,
    "rol_id": 3,
    "persona": {
      "documento": "12345678",
      "nombre": "María López",
      "correo": "maria@example.com",
      "id_sede": 1
    },
    "cargo": { "cargo_id": 2, "cargo_nombre": "Enfermero" },
    "rol": { "rol_id": 3, "rol_nombre": "Usuario" }
  }
]

```

- Campos y tipos esperables (recomendado para mostrar nombre junto a ID):
  - `id_emp`: number
  - `documento`: string
  - `id_dept`: number
  - `cargo_id`: number
  - `rol_id`: number
  - `persona`: object opcional con al menos `nombre` (string) y `documento` (string)
  - `cargo`: objeto opcional con `cargo_id` y `cargo_nombre`
  - `rol`: objeto opcional con `rol_id` y `rol_nombre`

`/empleados/{id}/:`

GET

```

{
  "id_emp": 23,
  "documento": "12345678",
  "id_dept": 4,
  "cargo_id": 2,
  "rol_id": 3,
  "persona": { "documento": "12345678", "nombre": "María López" },
  "cargo": { "cargo_id": 2, "cargo_nombre": "Enfermero" },
  "rol": { "rol_id": 3, "rol_nombre": "Usuario" }
}

```

POST `/empleados/` y PUT `/empleados/{id}/` usan/retornan estructuras similares (la capa de servicio mapea los campos entre API y frontend). Para que la UI muestre el nombre en el listado, asegúrate que `GET /empleados/` incluya un objeto `persona` por cada empleado con la propiedad `nombre`.

---

Módulo `personas` - `pacientes` (archivos:

`src/app/modules/personas/services/paciente.service.ts`, `pacientes-list`, `paciente-form`):

`/pacientes/:`

GET

```
[
  {
    "cod_pac": 502,
    "documento": "87654321",
    "persona": { "documento": "87654321", "nombre": "Ana Gómez" }
  }
]
```

- Campos y tipos:
  - `cod_pac`: number
  - `documento`: string
  - `persona`: object opcional con `nombre` (string)

`/pacientes/{id}/:`

GET

```
{
  "cod_pac": 502,
  "documento": "87654321",
  "persona": { "documento": "87654321", "nombre": "Ana Gómez" }
}
```

Para que la tabla muestre el nombre junto al ID, el endpoint `GET /pacientes/` debe regresar la propiedad `persona.nombre` o un campo `nombre` en el objeto paciente.

---

Módulo `historias` (archivos: `src/app/modules/historias/services/historia.service.ts`, `historias-list`, `historia-form`):

`/historias/:`

GET

```
[
  {
    "cod_hist": 1201,
    "fecha_registro_hora": "2025-11-10T14:30:00Z",
    "diagnostico": "Gripe",
    "cod_pac": 502,
    "id_emp": 23
  }
]
```

`/historias/{id}/:`

GET

```
{
  "cod_hist": 1201,
  "fecha_registro_hora": "2025-11-10T14:30:00Z",
  "diagnostico": "Gripe severa",
  "cod_pac": 502,
  "id_emp": 23,
  "prescripciones": [ /* opcional: array de medicamentos recetados */ ]
}
```

POST `/historias/` espera un payload similar a la estructura de creación (ej. `cod_pac`, `id_emp`, `diagnostico`, `fecha_registro_hora` opcional).

---

Si quieres, puedo:

- Ejecutar pruebas en caliente (levantar `npm start`) y probar las rutas UI para verificar que los botones ahora funcionen.
- Generar JSON Schema o ejemplos separados por endpoint.