Documentación del Frontend: Hospital Tracker

Descripción

Este proyecto es el frontend de la aplicación GeoAppHospital, encargado de la visualización y gestión de hospitales, empleados y monitoreo en tiempo real, utilizando React y Vite.

Instalación y ejecución

1. Instala las dependencias:

   ```
   npm install
   ```

2. Inicia el servidor de desarrollo:

   ```
   npm run dev
   ```

   La aplicación estará disponible en `http://localhost:5173` por defecto.

Estructura de carpetas principal

- **src/**: Código fuente principal.
  - **components/**: Componentes reutilizables y específicos por rol (admin, dashboard, etc).
  - **context/**: Contextos de React para manejo de estado global (ej. autenticación, ubicación).
  - **helpers/**: Funciones auxiliares.
  - **pages/**: Páginas principales de la app (Login, Dashboard, vistas por rol).
  - **config/**: Configuración de endpoints y utilidades.
  - **assets/**: Imágenes y recursos estáticos.
- **public/**: Archivos públicos y mapas.

Estructura basada en roles

La aplicación divide sus vistas y funcionalidades según el rol del usuario que inicia sesión. Esto permite que cada tipo de usuario acceda únicamente a las herramientas y datos relevantes para su función.

Roles disponibles

- **Superadmin**: Acceso total a la plataforma, gestión global de hospitales, usuarios y monitoreo.
- **Administrador estatal**: Gestión y monitoreo de hospitales y usuarios dentro de un estado específico.

- **Administrador municipal**: Gestión y monitoreo de hospitales y usuarios dentro de un municipio.
- **Administrador de hospital**: Gestión de información y personal de un hospital específico.

Cada rol tiene una vista principal y componentes específicos adaptados a sus necesidades.

Rutas principales

Las rutas están protegidas según el rol del usuario: - `/` — Login - `/dashboard` — Dashboard general - `/superadmin-geoapp` — Vista para superadministrador - `/estadoadmin-geoapp` — Vista para administrador estatal - `/municipioadmin-geoapp` — Vista para administrador municipal - `/hospitaladmin-geoapp` — Vista para administrador de hospital

Control de acceso

- El control de acceso se realiza mediante el componente `RoleProtectedRoute`.
- El archivo `App.jsx` centraliza la configuración de rutas y carga de páginas.

Superadmin: Detalle de Componentes y Endpoints

El rol **Superadmin** tiene acceso total a la plataforma y puede gestionar hospitales, administradores, grupos y empleados. A continuación se describen los principales componentes, los endpoints que utilizan y el flujo de trabajo para cada funcionalidad clave.

1. Lista de Hospitales

- **Componente:** `HospitalList` (`src/components/lists/HospitalList.jsx`)
- **Endpoint:** `GET /api/superadmin/hospitals`
- **Ejemplo de request:**

  ```
  GET https://geoapphospital-b0yr.onrender.com/api/superadmin/hospitals
  ```

- **Ejemplo de response:**


  ```
  [
    {
      "id_hospital": 1,
      "nombre_hospital": "Hospital General Central",
      "estado": "Baja California",
      "municipio": "Tijuana",
      "tipo_hospital": "General",
  ```


```
    "region": "Noroeste",
    "latitud_hospital": 32.5149,
    "longitud_hospital": -117.0382
  },
  ...
]
```

2. Stats Cards (Tarjetas de Estadísticas)

- **Componente:** `StatsCard` (`src/components/admin/StatsCard.jsx`)
- **Endpoints comunes:**
  - `GET /api/superadmin/hospitals` (total hospitales)
  - `GET /api/employees/get-empleados` (total empleados)
  - `GET /api/groups/get-groups` (total grupos)
- **Ejemplo de request para empleados:**

  ```
  GET https://geoapphospital-b0yr.onrender.com/api/employees/get-empleados
  ```

- **Ejemplo de response:**

```
[
  {
    "id_user": 123,
    "nombre": "Juan",
    "ap_paterno": "Pérez",
    "ap_materno": "López",
    "CURP": "PELJ800101HDFXXX01",
    "telefono": "1234567890",
    "role_name": "empleado",
    "id_hospital": 1,
    "id_grupo": 2
  },
  ...
]
```

3. Lista de Administradores

- **Componente:** `AdministradorList` (`src/components/lists/AdministradorList.jsx`)
- **Endpoint:** `GET /api/superadmin/get-admins`
- **Editar:** `PUT /api/superadmin/update-admins`
- **Eliminar:** `POST /api/superadmin/delete-admin/:id_user`
- **Ejemplo de request para editar:**

```
PUT https://geoapphospital-b0yr.onrender.com/api/superadmin/update-
admins
Content-Type: application/json

{
  "id_user": 123,
  "nombre": "Juan",
  "ap_paterno": "Pérez",
  "ap_materno": "López",
  "curp_user": "PELJ800101HDFXXX01"
}
```

- **Ejemplo de response:**

```
{ "success": true, "message": "Administrador actualizado" }
```

4. Lista de Grupos

- **Componente:** `GrupoList` (`src/components/lists/GrupoList.jsx`)
- **Endpoint:** `GET /api/groups/get-groups`
- **Crear:** `POST /api/groups/create-groups`
- **Editar:** `PUT /api/groups/update-groups`
- **Eliminar:** `POST /api/groups/delete-groups/:id_group`
- **Ejemplo de request para crear grupo:**

```
POST https://geoapphospital-b0yr.onrender.com/api/groups/create-gro
ups
Content-Type: application/json

{
  "nombre_grupo": "Limpieza",
  "descripcion_grupo": "Personal de limpieza",
  "id_hospital": 1
}
```

- **Ejemplo de response:**

```
{ "success": true, "id_group": 10 }
```

5. Lista de Empleados

- **Componente:** `EmpleadoList` (`src/components/lists/EmpleadoList.jsx`)
- **Endpoint:** `GET /api/employees/get-empleados`
- **Editar:** `PUT /api/employees/update-empleado`
- **Eliminar:** `POST /api/employees/delete-empleado/:id_user`

6. Agregar un nuevo empleado

- **Componente:** `EmpleadoForm` (`src/components/admin/EmpleadoForm.jsx`)
- **Endpoint:** `POST /api/employees/create-empleado-nombres`
- **Ejemplo de payload:**

```
{
  "nombre": "Juan",
  "ap_paterno": "Pérez",
  "ap_materno": "López",
  "CURP": "PELJ800101HDFXXX01",
  "correo_electronico": "juan.perez@correo.com",
  "telefono": "1234567890",
  "user": "jperez",
  "pass": "contraseñaGenerada",
  "role_name": "empleado",
  "id_estado": 2,
  "id_municipio": 5,
  "id_hospital": 1,
  "id_grupo": 3
}
```

- **Ejemplo de response:**

```
{ "success": true, "id_user": 456 }
```

7. Agregar empleados por CSV (Carga masiva)

- **Componente:** `SubidaMasivaEmpleados` (`src/components/admin/SubidaMasivaEmpleados.jsx`)
- **Endpoint:** `POST /api/employees/create-empleado-nombres`
- **Formato de archivo:** Excel (.xlsx) con columnas: ESTADO, MUNICIPIO, HOSPITAL, GRUPO, Nombre, Apellido Paterno, Apellido Materno, CURP, Telefono, Correo
- **Ejemplo de payload por cada fila:**

```
{
  "estado": "Baja California",
  "municipio": "Tijuana",
  "hospital": "Hospital General Central",
  "grupo": "Limpieza",
  "nombre": "Juan",
  "ap_paterno": "Pérez",
  "ap_materno": "López",
  "CURP": "PELJ800101HDFXXX01",
  "correo_electronico": "juan.perez@correo.com",
```

```
  "telefono": "1234567890",
  "user": "jperez",
  "pass": "contraseñaGenerada",
  "role_name": "empleado"
}
```

Notas de seguridad

- Todos los endpoints requieren autenticación mediante token JWT (Bearer Token) en el header `Authorization`.
- El rol superadmin es el único con acceso a los endpoints de gestión global.
- No compartas credenciales generadas por el sistema fuera de los canales oficiales.

Errores comunes y cómo resolverlos

- **401 Unauthorized:**
  - El token de autenticación es inválido o ha expirado. Solución: vuelve a iniciar sesión.
- **400 Bad Request:**
  - Algún campo obligatorio está vacío o mal formateado. Solución: revisa los datos enviados.
- **500 Internal Server Error:**
  - Error inesperado en el servidor. Solución: revisa la consola del backend y reporta el error.

Componentes clave de la vista Superadmin

A continuación se describen los componentes principales responsables de mostrar las listas de hospitales, administradores, grupos y empleados en la vista del Superadmin.

HospitalList

- **Ubicación:** `src/components/lists/HospitalList.jsx`
- **¿Qué hace?** Muestra la lista de hospitales registrados, permite filtrar por estado y paginar los resultados.
- **Props principales:**
  - `hospitales`: Array de hospitales a mostrar.
  - `estadoFiltro`: Estado seleccionado para filtrar.

    - `setEstadoFiltro`: Función para cambiar el filtro de estado.
    - `handleEditarHospital`: Función para editar un hospital.
    - `paginaActual`, `setPaginaActual`: Control de paginación.
- **Endpoint(s) que usa:**
    - Los datos se obtienen en el contenedor/página y se pasan como prop, normalmente usando `GET /api/superadmin/hospitals`.
- **Ejemplo de uso:**

```
<HospitalList
  hospitales={hospitales}
  estadoFiltro={estadoFiltro}
  setEstadoFiltro={setEstadoFiltro}
  handleEditarHospital={editarHospital}
  paginaActual={paginaActual}
  setPaginaActual={setPaginaActual}
/>
```

- **Notas adicionales:**
    - Internamente maneja paginación y filtrado.
    - No realiza fetch directamente, espera los datos como prop.

AdministradorList

- **Ubicación:** `src/components/lists/AdministradorList.jsx`
- **¿Qué hace?** Muestra la lista de administradores (superadmin, estatal, municipal, hospital), permite buscar, filtrar, editar y eliminar.
- **Props principales:**
    - `administradores`: Array de administradores a mostrar.
    - `tipoAdminFiltro`, `setTipoAdminFiltro`: Filtro por tipo de administrador.
    - `estadoAdminFiltro`, `setEstadoAdminFiltro`: Filtro por estado.
    - `busquedaAdmin`, `setBusquedaAdmin`: Búsqueda por nombre o CURP.
    - `onEditar`, `onEliminar`: Callbacks para editar/eliminar.
- **Endpoint(s) que usa:**
    - Los datos se obtienen en el contenedor/página y se pasan como prop, usando `GET /api/superadmin/get-admins`.
    - Acciones: `PUT /api/superadmin/update-admins`, `POST /api/superadmin/delete-admin/:id_user`.
- **Ejemplo de uso:**

```jsx
<AdministradorList
  administradores={admins}
  tipoAdminFiltro={tipoFiltro}
  setTipoAdminFiltro={setTipoFiltro}
  estadoAdminFiltro={estadoFiltro}
  setEstadoAdminFiltro={setEstadoFiltro}
  busquedaAdmin={busqueda}
  setBusquedaAdmin={setBusqueda}
  onEditar={editarAdmin}
  onEliminar={eliminarAdmin}
/>
```

- **Notas adicionales:**
    - Permite filtrar por tipo y estado, y búsqueda por nombre/CURP.
    - Internamente maneja modales para edición y eliminación.

GrupoList

- **Ubicación:** `src/components/lists/GrupoList.jsx`
- **¿Qué hace?** Muestra la lista de grupos de trabajo, permite filtrar por estado, buscar, editar y eliminar grupos.
- **Props principales:**
    - `grupos`: Array de grupos a mostrar.
    - `onGuardar`: Callback para actualizar la lista tras crear/editar/eliminar.
    - `hospitales`: Array de hospitales (opcional, para edición de grupo).
- **Endpoint(s) que usa:**
    - Los datos se obtienen en el contenedor/página y se pasan como prop, usando `GET /api/groups/get-groups`.
    - Acciones: `POST /api/groups/create-groups`, `PUT /api/groups/update-groups`, `POST /api/groups/delete-groups/:id_group`.
- **Ejemplo de uso:**

```jsx
<GrupoList
  grupos={grupos}
  onGuardar={actualizarGrupos}
  hospitales={hospitales}
/>
```

- **Notas adicionales:**
    - Permite búsqueda y filtrado por estado.

- Internamente maneja modales para edición y eliminación.

EmpleadoList

- **Ubicación:** `src/components/lists/EmpleadoList.jsx`
- **¿Qué hace?** Muestra la lista de empleados registrados, permite buscar, filtrar, editar y eliminar empleados.
- **Props principales:**
  - `empleados`: Array de empleados a mostrar.
  - `busquedaEmpleado`, `setBusquedaEmpleado`: Búsqueda por nombre o CURP.
  - `estadoEmpleadoFiltro`, `setEstadoEmpleadoFiltro`: Filtro por estado.
  - `onEmpleadosUpdate`: Callback para actualizar la lista tras cambios.
- **Endpoint(s) que usa:**
  - Los datos se obtienen en el contenedor/página y se pasan como prop, usando `GET /api/employees/get-empleados`.
  - Acciones: `PUT /api/employees/update-empleado`, `POST /api/employees/delete-empleado/:id_user`.
- **Ejemplo de uso:**

```
<EmpleadoList
  empleados={empleados}
  busquedaEmpleado={busqueda}
  setBusquedaEmpleado={setBusqueda}
  estadoEmpleadoFiltro={estadoFiltro}
  setEstadoEmpleadoFiltro={setEstadoFiltro}
  onEmpleadosUpdate={actualizarEmpleados}
/>
```

- **Notas adicionales:**
  - Permite búsqueda y filtrado por estado.
  - Internamente maneja modales para edición y eliminación.

Formularios de creación

A continuación se describen los formularios principales para la creación de hospitales, administradores, grupos y empleados en la vista del Superadmin.

HospitalForm

- **Ubicación:** `src/components/admin/HospitalForm.jsx`
- **¿Qué hace?** Permite crear un nuevo hospital, ingresando datos como nombre, estado, municipio, tipo, región y geocerca.
- **Props principales:**
    - `onGuardar`: Callback para guardar el hospital (el fetch al endpoint lo realiza el contenedor/página, no el formulario directamente).
    - `onCancelar`: Callback para cancelar la creación.
    - Otros props para edición y manejo de coordenadas/geocerca.
- **Endpoint sugerido:** `POST /api/superadmin/create-hospital` (usado en el contenedor/página)
- **Ejemplo de uso:**

```
<HospitalForm onGuardar={guardarHospital} onCancelar={cancelarHospi
tal} />
```

- **Ejemplo de payload:**

```
{
  "estado": "Baja California",
  "municipio": "Tijuana",
  "nombre": "Hospital General Central",
  "tipoUnidad": "General",
  "region": "Noroeste",
  "lat": 32.5149,
  "lng": -117.0382,
  "geocerca": { /* objeto GeoJSON */ }
}
```

- **Notas:** Valida que los campos sean correctos y que la geocerca esté definida.

AdminForm

- **Ubicación:** `src/components/admin/AdminForm.jsx`
- **¿Qué hace?** Permite crear nuevos administradores (superadmin, estatal, municipal, hospital).
- **Props principales:**
    - `hospitales`: Array de hospitales disponibles.
    - `onGuardar`: Callback para guardar el administrador (el fetch al endpoint lo realiza el contenedor/página, no el formulario directamente).

    - `onCancelar`: Callback para cancelar la creación.
    - Otros props para filtrado y edición.
- **Endpoint sugerido:** `POST /api/superadmin/create-admin` (usado en el contenedor/página)
- **Ejemplo de uso:**

```
<AdminForm hospitales={hospitales} onGuardar={guardarAdmin} onCance
lar={cancelarAdmin} />
```

- **Ejemplo de payload:**

```
{
  "nombre": "Juan",
  "ap_paterno": "Pérez",
  "ap_materno": "López",
  "CURP": "PELJ800101HDFXXX01",
  "telefono": "1234567890",
  "user": "jperez",
  "pass": "contraseñaGenerada",
  "role_name": "superadmin",
  "id_user_creador": 1
}
```

- **Notas:**
    - Genera usuario y contraseña automáticamente.
    - **Restricción importante:** Solo el superadmin con id 1 puede crear otros superadmins. Si intentas crear un superadmin con otro usuario, la opcion de crear superadmin no aparecera. Para crear un nuevo superadmin, debes iniciar sesión con el primer superadmin creado (id 1).

GrupoForm

- **Ubicación:** `src/components/admin/GrupoForm.jsx`
- **¿Qué hace?** Permite crear un nuevo grupo de trabajo asociado a un hospital.
- **Props principales:**
    - `onGuardar`: Callback para guardar el grupo.
    - `onCancelar`: Callback para cancelar la creación.
    - Otros props para edición y selección de hospital.
- **Endpoint:** `POST /api/groups/create-groups` (el fetch lo realiza el propio formulario).

- **Ejemplo de uso:**

```
<GrupoForm onGuardar={guardarGrupo} onCancelar={cancelarGrupo} />
```

EmpleadoForm

- **Ubicación:** `src/components/admin/EmpleadoForm.jsx`
- **¿Qué hace?** Permite crear un nuevo empleado, generando usuario y contraseña automáticamente.
- **Props principales:**
  - `onGuardar`: Callback para guardar el empleado (el fetch al endpoint lo realiza el contenedor/página, no el formulario directamente).
  - `onCancelar`: Callback para cancelar la creación.
  - Otros props para edición y selección de grupo/hospital.
- **Endpoint sugerido:** `POST /api/employees/create-empleado-nombres` (usado en el contenedor/página)
- **Ejemplo de uso:**

```
<EmpleadoForm onGuardar={guardarEmpleado} onCancelar={cancelarEmpleado} />
```

- **Notas:** Envía las credenciales por email al empleado.

SubidaMasivaEmpleados

- **Ubicación:** `src/components/admin/SubidaMasivaEmpleados.jsx`
- **¿Qué hace?** Permite la carga masiva de empleados mediante un archivo Excel.
- **Props principales:**
  - `onCancelar`: Callback para cancelar la carga masiva.
- **Endpoint:** `POST /api/employees/create-empleado-nombres` (el fetch lo realiza el propio formulario, uno por cada fila válida).
- **Ejemplo de uso:**

```
<SubidaMasivaEmpleados onCancelar={cancelarCargaMasiva} />
```

- **Notas:**

- El archivo debe tener las columnas: ESTADO, MUNICIPIO, HOSPITAL, GRUPO, Nombre, Apellido Paterno, Apellido Materno, CURP, Telefono, Correo.
- Valida los datos antes de enviar cada registro.

Monitoreo en tiempo real (Superadmin)

El rol **Superadmin** puede visualizar y monitorear la ubicación y actividad de los empleados en tiempo real, así como consultar el historial de movimientos y eventos relevantes en la plataforma.

1. Mapa de Monitoreo

- **Componente:** `MonitoreoMap` (`src/components/admin/MonitoreoMap.jsx`)
- **¿Qué hace?** Muestra un mapa interactivo con la ubicación en tiempo real de los empleados, agrupados por hospital, grupo o estado. Permite filtrar por estado, hospital, grupo y rango de fechas.
- **Props principales:**
  - `empleados`: Array de empleados a mostrar en el mapa.
  - `filtros`: Objeto con los filtros activos (estado, hospital, grupo, fechas).
  - `onFiltrar`: Callback para actualizar los filtros.
- **Endpoint(s) que usa:**
  - `GET /api/monitoreo/ubicaciones` (ubicaciones en tiempo real)
  - `GET /api/monitoreo/historial` (historial de ubicaciones)
- **Ejemplo de uso:**

```
<MonitoreoMap empleados={empleados} filtros={filtros} onFiltrar={ac
tualizarFiltros} />
```

- **Notas adicionales:**
  - Permite visualizar geocercas de hospitales y rutas recorridas.
  - Los datos se actualizan automáticamente cada cierto intervalo (ej. 30 segundos).

2. Configuracion de Monitoreo

- **Componente:** `MonitoreoConfig` (`src/components/admin/MonitoreoConfig.jsx`)

- **¿Qué hace?** Permite configurar los parámetros globales del monitoreo: niveles jerárquicos (estado, municipio, hospital, grupo), radio de geocerca, frecuencia de actualización y preferencias de alertas. Muestra una vista previa del monitoreo en tiempo real.
- **Incluye:**
    - Un formulario de configuración.
    - El componente `MonitoreoMap` embebido para visualizar la ubicación de empleados y hospitales en tiempo real.
- **Ejemplo de uso:**

  ```
  <MonitoreoConfig />
  ```

- **Notas:**
    - El monitoreo en tiempo real y la visualización de geocercas se realiza dentro de este componente.

Panel de actividad de administradores

El componente `ActivityLog` no es una bitácora de eventos del sistema ni de monitoreo general. Es un panel flotante que permite al superadmin visualizar y gestionar la asignación de trabajo de los administradores (superadmin, estatal, municipal, hospital), mostrando quién está activo, su ubicación y permitiendo cambiar su asignación de hospital, municipio o estado.

ActivityLog

- **Ubicación:** `src/components/ActivityLog.jsx`
- **¿Qué hace?** Muestra un panel lateral con la lista de administradores, su estatus (trabajando o sin asignar), su ubicación actual (estado, municipio, hospital) y permite al usuario editar su propia asignación de trabajo en tiempo real.
- **Props:** No recibe props, obtiene la información mediante fetch y contexto de usuario.
- **Endpoints que utiliza:**
    - `GET /api/superadmin/superadmin-hospital-ubi/:id_user` (información del superadmin actual)
    - `GET /api/superadmin/superadmin-hospital-work` (lista de todos los administradores y su asignación)
    - `PUT /api/superadmin/superadmin-hospital` (actualizar asignación de trabajo)

    - `GET /api/superadmin/estados`, `GET /api/municipioadmin/municipios-by-estado-hospital/:id_estado`, `GET /api/superadmin/superadmin-hospitals-by-municipio` (para cargar opciones de ubicación)
- **Ejemplo de uso:** El componente se monta automáticamente y muestra un botón flotante en la interfaz. Al hacer clic, despliega el panel de administradores activos.

  ```
  <ActivityLog />
  ```

- **Notas:**
    - Permite al usuario cambiar su propio estado de trabajo (trabajando/terminado) y reasignarse a otro hospital disponible.
    - El panel muestra el número de administradores activos y su ubicación actual.
    - No es una bitácora de eventos ni de monitoreo de empleados.

Dashboards de Monitoreo

El componente principal para la visualización de dashboards es `MonitoreoDashboard`, ubicado en `src/components/dashboard/MonitoreoDashboard.jsx`. Este componente permite navegar entre diferentes niveles de análisis (nacional, estatal, municipal y hospitalario) mediante un sistema de pestañas visuales.

- **Componente:** `MonitoreoDashboard` (`src/components/dashboard/MonitoreoDashboard.jsx`)
- **¿Qué hace?** Muestra un panel de navegación con pestañas para seleccionar el nivel de dashboard a visualizar. Según la pestaña seleccionada, renderiza el dashboard correspondiente: nacional, estatal, municipal o hospital.
- **Niveles disponibles:**
    - **Nacional:** Vista general del país.
    - **Estatal:** Análisis por estado.
    - **Municipal:** Detalle municipal.
    - **Hospital:** Información hospitalaria.
- **Props principales:**
  No recibe props externos; maneja su propio estado interno para la pestaña seleccionada y, en el caso municipal, el estado y municipio activos.
- **Componentes hijos:**

    - `NacionalDashboard` (`src/components/dashboard/nacional/NacionalDashboard.jsx`)
    - `EstatalDashboard` (`src/components/dashboard/estatal/EstatalDashboard.jsx`)
    - `MunicipalDashboard` (`src/components/dashboard/municipal/MunicipalDashboard.jsx`)
    - `HospitalDashboard` (`src/components/dashboard/hospital/HospitalDashboard.jsx`)
- **Ejemplo de uso:**

  ```
  <MonitoreoDashboard />
  ```

- **Notas adicionales:**
    - El usuario puede cambiar de nivel de análisis haciendo clic en las pestañas del encabezado.
    - Cada dashboard muestra información y visualizaciones específicas para su nivel.
    - El diseño es responsivo y utiliza estilos modernos para una experiencia clara y visual.

Dashboard Nacional

- **Ubicación:**
    - `src/components/dashboard/nacional/NacionalDashboard.jsx`
    - **Generador de pdf:** `src/components/dashboard/nacional/reportes/NacionalReportPDF.jsx`

¿Cómo se usa? El dashboard nacional es el primer nivel que se muestra al renderizar `<MonitoreoDashboard />`. Se selecciona automáticamente cuando el usuario entra a la vista de dashboards, o al hacer clic en la pestaña “Nacional”.

```
import MonitoreoDashboard from "src/components/dashboard/MonitoreoDashboard";
// ...
<MonitoreoDashboard />
```

Internamente, el componente `MonitoreoDashboard` renderiza `<NacionalDashboard />` cuando la pestaña activa es “Nacional”.

Endpoints principales usados: - `/api/dashboards/nacional/estadisticas-estados?fechaInicio=...&fechaFin=...` - `/api/dashboards/nacional/totales?fechaInicio=...&fechaFin=...`

Cálculo de horas: - Las horas totales trabajadas se obtienen del backend vía el endpoint `/api/dashboards/nacional/totales`. - El backend entrega el total de horas ya calculado para el periodo seleccionado.

Componentes principales internos: - Mapas interactivos de México (`react-simple-maps`) para mostrar datos por estado. - Tarjetas de KPIs (hospitales, empleados, salidas de geocerca, horas trabajadas). - Listas laterales de Top 10 estados (por salidas y por horas). - Selector de fechas y presets de rango. - Botón para descargar reporte PDF nacional.

Flujo y estructura: 1. El usuario puede seleccionar el rango de fechas a analizar (presets rápidos o personalizado). 2. Se muestran KPIs nacionales (hospitales, empleados, salidas, horas). 3. Se despliegan dos mapas de calor: - Salidas de geocerca por estado (color rojo) - Horas trabajadas por estado (color azul) 4. A la derecha de cada mapa, se muestra el Top 10 de estados según el indicador correspondiente. 5. Al pasar el mouse sobre un estado, aparece un tooltip con datos detallados de ese estado. 6. El usuario puede descargar un PDF con el resumen nacional.

Dashboard Estatal

- **Ubicación:**
    - `src/components/dashboard/estatal/EstatalDashboard.jsx`
    - Generador de PDF: `src/components/dashboard/estatal/reportes/EstatalReportPDF.jsx`
- **¿Cómo se usa?** El dashboard estatal se muestra al seleccionar la pestaña “Estatal” en `<MonitoreoDashboard />`.
    - Si el usuario es **superadmin**, puede seleccionar cualquier estado desde el selector.
    - Si el usuario es **administrador estatal** (`estadoadmin`), el estado se detecta automáticamente por el contexto de usuario y el selector queda bloqueado.
- **Componentes principales internos:**
    - Mapas interactivos de municipios del estado seleccionado (`react-simple-maps`, `topojson-client`, `d3-geo`).
    - Tarjetas de KPIs estatales (hospitales, empleados, salidas de geocerca, horas trabajadas, horas fuera, horas descanso).
    - Gráficas de barras, líneas y áreas (`recharts`).
    - Rankings de hospitales y municipios.
    - Tooltips detallados por municipio.
    - Botón para descargar reporte PDF estatal (`generarReporteEstatalPDF`).
- **Flujo y estructura:**

1. Selección de estado (o asignación automática por rol) y rango de fechas.
2. Carga y filtrado de municipios usando TopoJSON (`mx_tj.json`).
3. Render de mapas, KPIs, gráficas y rankings.
4. Descarga de PDF estatal.

- **Endpoints principales usados:**
  - `/api/dashboards/estatal/entradas-salidas`
  - `/api/dashboards/estatal/eventos-geocerca`
  - `/api/dashboards/estatal/ranking-hospitales`
  - `/api/dashboards/estatal/horas-municipio`
  - `/api/dashboards/estatal/metricas`
  - `/api/dashboards/estatal/distribucion-municipal-completa` (o `/distribucion-municipal` como fallback)
  - `/api/dashboards/estatal/municipio-detalle`
  - `/api/dashboards/municipios-by-estado/:id_estado`
  - Para admins estatales: `/api/estadoadmin/hospitals-by-user/:userId` (para saber el estado asignado)
- **Cálculo de horas:**
  - Las horas trabajadas, horas fuera y horas de descanso se obtienen desde el endpoint `/api/dashboards/estatal/metricas?id_estado=...&fechaInicio=...&fechaFin=....`
  - El cálculo es realizado por el backend y se muestra en KPIs y gráficas.
  - Para cada municipio, las horas se obtienen de `/api/dashboards/estatal/horas-municipio?id_estado=...&fechaInicio=...&fechaFin=....`
- **Props relevantes:** No recibe props externos. Todo el fetch y estado es interno.
- **Ejemplo de uso directo:**

```
import EstatalDashboard from "src/components/dashboard/estatal/EstatalDashboard";
<EstatalDashboard />
```

- **Notas:**
  - El componente espera que los archivos `/lib/mx_tj.json` y `/lib/state_map_config.json` estén disponibles en `public/`.
  - El PDF se genera usando una función auxiliar incluida en el repo.
  - El diseño es responsivo y visualmente atractivo.
  - El comportamiento del selector de estado depende del rol del usuario.

- **Props relevantes:** No recibe props externos. Todo el fetch y estado es interno.
- **Ejemplo de uso directo:**

```
import NacionalDashboard from "src/components/dashboard/nacional/NacionalDashboard";
<NacionalDashboard />
```

- **Notas:**
    - El componente espera que el archivo de geografía `/lib/mx.json` esté disponible en `public/`.
    - El PDF se genera usando una función auxiliar (no incluida en el repo actual).
    - El diseño es responsivo y visualmente atractivo.

Dashboard Municipal

- **Ubicación:**
    - `src/components/dashboard/municipal/MunicipalDashboard.jsx`
    - Generador de PDF: `src/components/dashboard/municipal/reportes/MunicipalReportPDF.jsx`
- **¿Cómo se usa?** El dashboard municipal se muestra al seleccionar la pestaña "Municipal" en `<MonitoreoDashboard />`.
    - Si el usuario es **superadmin**, puede seleccionar cualquier estado y municipio.
    - Si el usuario es **administrador estatal** (`estadoadmin`), puede seleccionar cualquier municipio de su estado asignado.
    - Si el usuario es **administrador municipal** (`municipioadmin`), el municipio y estado se detectan automáticamente por el contexto de usuario y los selectores quedan bloqueados.
- **Componentes principales internos:**
    - Mapa interactivo del municipio seleccionado (`react-leaflet`, `topojson-client`).
    - Tarjetas de KPIs municipales (hospitales, empleados, salidas de geocerca, horas trabajadas, horas fuera, horas descanso).
    - Gráficas de barras, líneas, áreas y pastel (`recharts`).
    - Listado de hospitales y empleados del municipio.
    - Tooltips detallados por hospital.
    - Botón para descargar reporte PDF municipal (`generarReporteMunicipalPDF`).

- **Flujo y estructura:**
    1. Selección de estado y municipio (o asignación automática por rol) y rango de fechas.
    2. Carga de datos del municipio y hospitales vía endpoint.
    3. Render de mapa, KPIs, gráficas y listados.
    4. Descarga de PDF municipal.
- **Endpoints principales usados:**
    - `GET /api/superadmin/estados`
    - `GET /api/municipioadmin/municipios-by-estado-hospital/:id_estado`
    - `POST /api/dashboards/municipio` (con `{ id_municipio, fechaInicio, fechaFin }`)
    - Para admins estatales: `GET /api/estadoadmin/hospitals-by-user/:userId`
    - Para admins municipales: `GET /api/municipioadmin/hospitals-by-user/:userId`
- **Cálculo de horas:**
    - El endpoint `/api/dashboards/municipio` regresa los registros de empleados y hospitales del municipio.
    - El cálculo de horas trabajadas, fuera, descanso y salidas por hospital se realiza en frontend usando las funciones `calcularEstadisticasEmpleado` y `calcularEstadisticasEmpleadoPorDias` (`src/components/dashboard/hospital/employeeStatsHelper.js`).
- **Props relevantes:** No recibe props externos. Todo el fetch y estado es interno.
- **Ejemplo de uso directo:**

```
import MunicipalDashboard from "src/components/dashboard/municipal/
MunicipalDashboard";
<MunicipalDashboard />
```

- **Notas:**
    - El componente espera que los archivos `/lib/mx_tj.json` y `/lib/mx.json` estén disponibles en `public/`.
    - El PDF se genera usando una función auxiliar incluida en el repo.
    - El diseño es responsivo y visualmente atractivo.
    - El comportamiento de los selectores depende del rol del usuario.

Dashboard Hospitalario

- **Ubicación:**

    - `src/components/dashboard/hospital/HospitalDashboard.jsx` (componente principal)
    - `src/components/dashboard/hospital/Empleadodashboard.jsx` (EmpleadoDashboard: vista y lógica de empleados, usado internamente)
    - `src/components/dashboard/hospital/Grupodashboard.jsx` (GrupoDashboard: vista y lógica de grupos, usado internamente)
    - `src/components/dashboard/hospital/reportes/EmployeeReportPDF.jsx` (generador de PDF individual de empleado)
    - `src/components/dashboard/hospital/reportes/GroupReportPDF.jsx` (generador de PDF grupal de grupo)
- **¿Cómo se usa?** El dashboard hospitalario se muestra al seleccionar la pestaña "Hospital" en la vista de monitoreo o al acceder como administrador de hospital.
    - Si el usuario es **superadmin**, puede seleccionar cualquier estado, municipio y hospital.
    - Si el usuario es **administrador de hospital** (`hospitaladmin`), el hospital, municipio y estado se detectan automáticamente por el contexto de usuario y los selectores quedan bloqueados.
- Selector de estado, municipio y hospital (bloqueados según rol).
- Tarjetas de KPIs hospitalarios (grupos, empleados, salidas de geocerca, horas trabajadas).
- Gráficas de barras y líneas por grupo y empleado (`recharts`).
- Listado de empleados y grupos del hospital.
- Calendario de asistencia por empleado con desglose diario y por hora.
- Botón para descargar reporte PDF individual de empleado (`generarReporteEmpleadoPDF`) y de grupo (`generarReporteGrupoPDF`).
- **EmpleadoDashboard** (`src/components/dashboard/hospital/Empleadodashboard.jsx`): Renderiza la vista detallada y lógica de empleados, incluyendo calendario, KPIs y generación de PDF individual. Se utiliza internamente por `HospitalDashboard`.
- **GrupoDashboard** (`src/components/dashboard/hospital/Grupodashboard.jsx`): Renderiza la vista detallada y lógica de grupos, incluyendo KPIs y generación de PDF grupal. Se utiliza internamente por `HospitalDashboard`.

- **EmployeeReportPDF.jsx** y **GroupReportPDF.jsx**: Componentes auxiliares para la generación de reportes PDF de empleados y grupos, invocados desde los dashboards correspondientes.

- **Flujo y estructura:**

    1. Selección de estado, municipio y hospital (o asignación automática por rol) y rango de fechas.
    2. Carga de empleados y grupos vía endpoint.
    3. Carga de registros de actividad por empleado vía endpoint.
    4. Render de KPIs, gráficas, listados y calendario.
    5. Descarga de PDF individual o grupal.

- **Endpoints principales usados:**

    - `GET /api/superadmin/estados`
    - `GET /api/municipioadmin/municipios-by-estado-hospital/:id_estado`
    - `GET /api/hospitaladmin/hospitals-by-municipio?id_estado=:id_estado&id_municipio=:id_municipio`
    - `GET /api/hospitaladmin/empleados-by-ubicacion?id_hospital=:id_hospital`
    - `POST /api/reportes/empleado` (con `{ empleadoId, fechaInicio, fechaFin }`)
    - `POST /api/dashboards/grupo` (con `{ id_hospital, fechaInicio, fechaFin }`)
    - Para admins hospitalarios: `GET /api/hospitaladmin/hospital-by-user/:userId`

- **Cálculo de horas:**

    - El endpoint `/api/reportes/empleado` regresa los registros de actividad de cada empleado.
    - El cálculo de horas trabajadas, fuera, descanso y salidas por empleado se realiza en frontend usando las funciones `calcularEstadisticasEmpleado` y `calcularEstadisticasEmpleadoPorDias` (`src/components/dashboard/hospital/employeeStatsHelper.js`).

- **Props relevantes:** No recibe props externos. Todo el fetch y estado es interno.

- **Ejemplo de uso directo:**

```
import HospitalDashboard from "src/components/dashboard/hospital/HospitalDashboard";
<HospitalDashboard />
```

- **Notas:**

- El componente espera que los archivos `/lib/mx_tj.json` y `/lib/mx.json` estén disponibles en `public/`.
- El PDF se genera usando funciones auxiliares incluidas en el repo.
- El comportamiento de los selectores depende del rol del usuario.
- El calendario de asistencia por empleado permite ver detalles diarios y por hora, así como descargar el reporte PDF.

Administrador estatal

Dashboards y Monitoreo

El administrador estatal también tiene acceso a los dashboards y al monitoreo, pero con las siguientes particularidades:

- **EstatalDashboard** (`src/components/dashboard/estatal/EstatalDashboard.jsx`): Se utiliza para mostrar el dashboard estatal, pero el selector de estado está bloqueado y solo muestra el estado asignado al administrador.
- **MonitoreoDashboard** (`src/components/dashboard/MonitoreoDashboard.jsx`): Se reutiliza el dashboard de monitoreo general, pero el estado se autocompleta y bloquea según el usuario autenticado. El administrador estatal solo puede ver información de su propio estado.

Esto significa que, aunque la interfaz es similar a la del superadmin, el administrador estatal no puede cambiar de estado ni ver información de otros estados. Todos los datos y visualizaciones se filtran automáticamente para su estado asignado.

La vista de Administrador estatal permite gestionar y monitorear hospitales, grupos y empleados dentro del estado asignado. Incluye navegación lateral, KPIs, listados y filtros avanzados. > **Nota importante:** El administrador estatal solo tiene permisos de visualización. Puede consultar la información de hospitales, grupos y empleados de su estado, pero no puede crear, editar ni eliminar registros, ni agregar empleados. Todas las acciones son solo de consulta.

EstadoSidebar.jsx

- **Ubicación:**
  `src/components/estado/EstadoSidebar.jsx`

- **¿Qué hace?**
  Barra lateral de navegación para la vista de administrador estatal. Permite cambiar entre las secciones de hospitales, grupos, empleados, monitoreo y dashboard. Incluye íconos y animaciones de apertura/cierre.

- **¿Cómo se usa?**
  Se renderiza en la vista principal del administrador estatal y controla la pestaña activa mediante props.

- **Componentes principales internos:**
  - Botones de navegación con íconos (`lucide-react`).
  - Control de apertura/cierre de la barra lateral.
- **Flujo y estructura:**
  1. Muestra las opciones de navegación.
  2. Cambia la pestaña activa al hacer clic en una opción.
  3. Permite cerrar la sesión desde la barra.
- **Props relevantes:**
  - `activeTab`, `setActiveTab`: Controlan la pestaña activa.
  - `sidebarOpen`, `setSidebarOpen`: Controlan la apertura/cierre de la barra.
- **Ejemplo de uso directo:**

```
import EstadoSidebar from "src/components/estado/EstadoSidebar";
<EstadoSidebar activeTab={tab} setActiveTab={setTab} sidebarOpen={o
pen} setSidebarOpen={setOpen} />
```

- **Notas:**
  - Usa el contexto de autenticación para cerrar sesión.
  - El diseño es responsivo y moderno.

GrupoListEstado.jsx

- **Ubicación:**
  `src/components/estado/GrupoListEstado.jsx`

- **¿Qué hace?**
  Lista y filtra los grupos de trabajo del estado. Permite buscar grupos por nombre, filtrar por municipio y muestra KPIs de grupos, hospitales y empleados.

- **¿Cómo se usa?**
  Se utiliza en la vista de administrador estatal para consultar y filtrar grupos del estado asignado.

- **Componentes principales internos:**
  - Usa `StatsCardEstado` para mostrar KPIs.
  - Filtros de búsqueda y municipio.

    - Agrupación de grupos por municipio.
- **Flujo y estructura:**
    1. Obtiene el usuario autenticado y carga grupos vía endpoint.
    2. Muestra KPIs generales del estado.
    3. Permite buscar grupos y filtrar por municipio.
    4. Agrupa y muestra los grupos por municipio.
- **Endpoints principales usados:**
    - `GET /api/estadoadmin/groups-by-user/:id_user?source=groups` (lista de grupos)
    - `GET /api/estadoadmin/stats-by-user/:id_user?source=stats` (KPIs generales)
- **Cálculo de métricas:**
  Los KPIs se obtienen del backend y se muestran con `StatsCardEstado`.
- **Props relevantes:**
    - `id_user`: ID del usuario autenticado (admin estatal).
- **Ejemplo de uso directo:**

```
import GrupoListEstado from "src/components/estado/GrupoListEstado"
;
<GrupoListEstado id_user={userId} />
```

- **Notas:**
    - Incluye animaciones y feedback visual.
    - El diseño es responsivo y moderno.

HospitalListEstado.jsx

- **Ubicación:**
  `src/components/estado/HospitalListEstado.jsx`
- **¿Qué hace?**
  Muestra la lista de hospitales del estado, con paginación y KPIs. Permite filtrar hospitales y ver métricas generales.
- **¿Cómo se usa?**
  Se utiliza en la vista de administrador estatal para consultar hospitales del estado asignado.
- **Componentes principales internos:**
    - Usa `StatsCardEstado` para mostrar KPIs.

    - Paginación de hospitales.
- **Flujo y estructura:**
    1. Obtiene el usuario autenticado y carga hospitales vía endpoint.
    2. Muestra KPIs generales del estado.
    3. Permite paginar la lista de hospitales.
- **Endpoints principales usados:**
    - `GET /api/estadoadmin/stats-by-user/:id_user?source=stats` (KPIs generales)
- **Cálculo de métricas:**
  Los KPIs se obtienen del backend y se muestran con `StatsCardEstado`.
- **Props relevantes:**
    - `estadoNombre`, `hospitales`, `loading`, `municipioNombre`: Props para mostrar y filtrar hospitales.
- **Ejemplo de uso directo:**

```
import HospitalListEstado from "src/components/estado/HospitalListEstado";
<HospitalListEstado hospitales={hospitales} />
```

- **Notas:**
    - El diseño es responsivo y moderno.
    - Muestra mensajes si no hay hospitales registrados.

StatsCardEstado.jsx

- **Ubicación:**
  `src/components/estado/StatsCardEstado.jsx`
- **¿Qué hace?**
  Tarjeta visual para mostrar KPIs/estadísticas de hospitales, grupos o empleados, con ícono, color y subtítulo.
- **¿Cómo se usa?**
  Se utiliza en los listados y dashboards estatales para mostrar métricas clave.
- **Componentes principales internos:**
    - No tiene subcomponentes, es una tarjeta visual reutilizable.
- **Flujo y estructura:**
    1. Recibe props de ícono, label, valor, color y subtítulo.

2. Renderiza la tarjeta con estilos y colores según el tipo de KPI.

- **Endpoints principales usados:**
  No consume endpoints directamente.

- **Props relevantes:**

  - `icon`, `label`, `value`, `color`, `subtitle`.

- **Ejemplo de uso directo:**

```
<StatsCardEstado icon={<Hospital />} label="Hospitales" value={10}
color="emerald" subtitle="Hospitales registrados" />
```

- **Notas:**

  - Soporta varios colores y estilos.
  - Es reutilizable en cualquier vista estatal.

Administrador municipal

Dashboards y Monitoreo

El administrador municipal también tiene acceso al dashboard municipal y al mapa de monitoreo, pero con las siguientes particularidades:

- **MunicipalDashboard** (`src/components/dashboard/municipal/MunicipalDashboard.jsx`): Se utiliza para mostrar el dashboard municipal, pero el selector de estado y municipio está bloqueado y solo muestra el municipio asignado al administrador.
- **MonitoreoMap** (`src/components/admin/MonitoreoMap.jsx`): Se reutiliza el mapa de monitoreo general, pero el municipio y estado se autocompletan y bloquean según el usuario autenticado. El administrador municipal solo puede ver información de su propio municipio.

Esto significa que, aunque la interfaz es similar a la de otros roles, el administrador municipal no puede cambiar de municipio ni ver información de otros municipios o estados. Todos los datos y visualizaciones se filtran automáticamente para su municipio asignado.

> **Nota importante:** El administrador municipal solo tiene permisos de visualización. Puede consultar la información de hospitales, grupos y empleados de su municipio, así como acceder al dashboard y monitoreo, pero no puede crear, editar ni eliminar registros. Todas las acciones son solo de consulta.

EmpleadoListMunicipio.jsx

- **Ubicación:** `src/components/municipal/EmpleadoListMunicipio.jsx`

- **¿Qué hace?** Muestra la lista de empleados del municipio asignado al administrador municipal. Permite buscar empleados, ver estadísticas y filtrar resultados. Solo visualización, sin permisos de edición, creación ni eliminación.
- **Props principales:**
    - `id_user`: ID del usuario actual (admin municipal), usado para filtrar y obtener datos.
- **Endpoints que consume:**
    - `GET /api/municipioadmin/empleados-by-user/:id_user` — Lista de empleados del municipio.
    - `GET /api/municipioadmin/stats-by-user/:id_user?source=stats` — Estadísticas generales (KPIs) del municipio.
- **Lógica de cálculo:**
    - Realiza fetch de empleados y estadísticas al montar el componente o cambiar el usuario.
    - Implementa búsqueda con debounce profesionalizado para filtrar empleados localmente.
- **Notas sobre permisos:**
    - El administrador municipal solo puede visualizar la información, no puede crear, editar ni eliminar empleados.
- **Ejemplo de uso:**

```
<EmpleadoListMunicipio id_user={userId} />
```

- **Notas adicionales:**
    - Muestra mensajes de carga y error.
    - Utiliza `StatsCardMunicipio` para mostrar KPIs.

GrupoListMunicipio.jsx

- **Ubicación:** `src/components/municipal/GrupoListMunicipio.jsx`
- **¿Qué hace?** Muestra la lista de grupos del municipio asignado. Permite buscar grupos, ver estadísticas y filtrar resultados. Solo visualización, sin permisos de edición, creación ni eliminación.
- **Props principales:**
    - `id_user`: ID del usuario actual (admin municipal), usado para obtener los grupos.

- **Endpoints que consume:**
    - `GET /api/municipioadmin/grupos-by-user/:id_user?source=groups` — Lista de grupos y municipios asociados.
    - `GET /api/municipioadmin/stats-by-user/:id_user?source=stats` — Estadísticas generales del municipio.
- **Lógica de cálculo:**
    - Obtiene grupos y estadísticas al montar el componente.
    - Implementa búsqueda con debounce profesionalizado para filtrar grupos localmente.
- **Notas sobre permisos:**
    - El administrador municipal solo puede visualizar la información, no puede crear, editar ni eliminar grupos.
- **Ejemplo de uso:**

```jsx
<GrupoListMunicipio id_user={userId} />
```

- **Notas adicionales:**
    - Muestra mensajes de carga y error.
    - Utiliza `StatsCardMunicipio` para mostrar KPIs.

HospitalListMunicipio.jsx

- **Ubicación:** `src/components/municipal/HospitalListMunicipio.jsx`
- **¿Qué hace?** Muestra la lista de hospitales del municipio asignado. Permite buscar hospitales, ver estadísticas y paginar resultados. Solo visualización, sin permisos de edición, creación ni eliminación.
- **Props principales:**
    - `estadoNombre`: Nombre del estado (opcional, para mostrar contexto).
    - `hospitales`: Array de hospitales a mostrar.
    - `loading`: Estado de carga.
    - `municipioNombre`: Nombre del municipio (opcional).
- **Endpoints que consume:**
    - `GET /api/municipioadmin/stats-by-user/:id_user?source=stats` — Estadísticas generales del municipio.
- **Lógica de cálculo:**
    - Filtra hospitales localmente según búsqueda.
    - Implementa paginación y búsqueda con debounce.

- **Notas sobre permisos:**
    - El administrador municipal solo puede visualizar la información, no puede crear, editar ni eliminar hospitales.
- **Ejemplo de uso:**

```
<HospitalListMunicipio hospitales={hospitales} municipioNombre={mun
icipio} />
```

- **Notas adicionales:**
    - Muestra mensajes si no hay hospitales registrados.
    - Utiliza `StatsCardMunicipio` para mostrar KPIs.

MunicipalSidebar.jsx

- **Ubicación:** `src/components/municipal/MunicipalSidebar.jsx`
- **¿Qué hace?** Barra lateral de navegación para la vista de administrador municipal. Permite cambiar entre pestañas de hospitales, grupos, empleados, monitoreo y dashboard.
- **Props principales:**
    - `activeTab`, `setActiveTab`: Controlan la pestaña activa.
    - `sidebarOpen`, `setSidebarOpen`: Controlan la apertura/cierre del sidebar.
- **Endpoints que consume:**
    - No consume endpoints directamente.
- **Notas sobre permisos:**
    - Solo navegación, no permite acciones de edición ni creación.
- **Ejemplo de uso:**

```
<MunicipalSidebar activeTab={activeTab} setActiveTab={setActiveTab}
sidebarOpen={sidebarOpen} setSidebarOpen={setSidebarOpen} />
```

- **Notas adicionales:**
    - Muestra íconos y nombres de cada sección.
    - El diseño es responsivo y moderno.

StatsCardMunicipio.jsx

- **Ubicación:** `src/components/municipal/StatsCardMunicipio.jsx`

- **¿Qué hace?** Tarjeta visual para mostrar KPIs/estadísticas de hospitales, grupos o empleados a nivel municipal, con ícono, color y subtítulo.
- **Props principales:**
  - `icon`, `label`, `value`, `color`, `subtitle`.
- **Endpoints que consume:**
  - No consume endpoints directamente; recibe datos por props.
- **Notas sobre permisos:**
  - Solo visualización.
- **Ejemplo de uso:**

```
<StatsCardMunicipio icon={<Hospital />} label="Hospitales" value={5
} color="emerald" subtitle="Hospitales registrados" />
```

- **Notas adicionales:**
  - Soporta varios colores y estilos.
  - Es reutilizable en cualquier vista municipal.

Administrador de hospital

El administrador de hospital tiene acceso al dashboard hospitalario y al monitoreo, pero únicamente con los datos del hospital que tiene asignado. No puede ver ni seleccionar otros hospitales, municipios o estados. Todos los datos, KPIs y visualizaciones se filtran automáticamente para su hospital asignado.

> **Nota importante:** El administrador de hospital solo tiene permisos de visualización y consulta sobre su hospital, empleados y grupos. No puede crear, editar ni eliminar registros. Todas las acciones son solo de consulta.

Dashboards y Monitoreo

- **HospitalDashboard** (`src/components/dashboard/hospital/HospitalDashboard.jsx`): Se utiliza para mostrar el dashboard hospitalario, pero el selector de estado, municipio y hospital está bloqueado y solo muestra el hospital asignado al administrador.
- **MonitoreoMap** (`src/components/admin/MonitoreoMap.jsx`): Se reutiliza el mapa de monitoreo general, pero el hospital, municipio y estado se autocompletan y bloquean según el usuario autenticado. El administrador de hospital solo puede ver información de su propio hospital.

Esto significa que, aunque la interfaz es similar a la de otros roles, el administrador de hospital no puede cambiar de hospital ni ver información de otros hospitales,

municipios o estados. Todos los datos y visualizaciones se filtran automáticamente para su hospital asignado.

EmpleadoListHospital.jsx

- **Ubicación:** `src/components/hospital/EmpleadoListHospital.jsx`
- **¿Qué hace?** Muestra la lista de empleados del hospital asignado al administrador de hospital. Permite buscar empleados, ver estadísticas y filtrar resultados. Solo visualización, sin permisos de edición, creación ni eliminación.
- **Props principales:**
  - `id_user`: ID del usuario actual (admin hospital), usado para filtrar y obtener datos.
- **Endpoints que consume:**
  - `GET /api/hospitaladmin/empleados-by-hospital/:hospitalId` — Lista de empleados del hospital.
  - `GET /api/hospitaladmin/stats-by-hospital/:hospitalId?source=stats` — Estadísticas generales (KPIs) del hospital.
- **Lógica de cálculo:**
  - Realiza fetch de empleados y estadísticas al montar el componente o cambiar el usuario.
  - Implementa búsqueda con debounce profesionalizado para filtrar empleados localmente.
- **Notas sobre permisos:**
  - El administrador de hospital solo puede visualizar la información, no puede crear, editar ni eliminar empleados.
- **Ejemplo de uso:**

```jsx
<EmpleadoListHospital id_user={userId} />
```

- **Notas adicionales:**
  - Muestra mensajes de carga y error.
  - Utiliza `StatsCardHospital` para mostrar KPIs.

GrupoListHospital.jsx

- **Ubicación:** `src/components/hospital/GrupoListHospital.jsx`

- **¿Qué hace?** Muestra la lista de grupos del hospital asignado. Permite buscar grupos, ver estadísticas y filtrar resultados. Solo visualización, sin permisos de edición, creación ni eliminación.
- **Props principales:**
  - `id_user`: ID del usuario actual (admin hospital), usado para obtener los grupos.
- **Endpoints que consume:**
  - `GET /api/hospitaladmin/grupos-by-hospital/:hospitalId` — Lista de grupos del hospital.
  - `GET /api/hospitaladmin/stats-by-hospital/:hospitalId?source=stats` — Estadísticas generales del hospital.
- **Lógica de cálculo:**
  - Obtiene grupos y estadísticas al montar el componente.
  - Implementa búsqueda con debounce profesionalizado para filtrar grupos localmente.
- **Notas sobre permisos:**
  - El administrador de hospital solo puede visualizar la información, no puede crear, editar ni eliminar grupos.
- **Ejemplo de uso:**

```
<GrupoListHospital id_user={userId} />
```

- **Notas adicionales:**
  - Muestra mensajes de carga y error.
  - Utiliza `StatsCardHospital` para mostrar KPIs.

HospitalAdminSidebar.jsx

- **Ubicación:** `src/components/hospital/HospitalAdminSidebar.jsx`
- **¿Qué hace?** Barra lateral de navegación para la vista de administrador de hospital. Permite cambiar entre pestañas de grupos, empleados, monitoreo y dashboard.
- **Props principales:**
  - `activeTab`, `setActiveTab`: Controlan la pestaña activa.
  - `sidebarOpen`, `setSidebarOpen`: Controlan la apertura/cierre del sidebar.

- **Endpoints que consume:**
  - No consume endpoints directamente.
- **Notas sobre permisos:**
  - Solo navegación, no permite acciones de edición ni creación.
- **Ejemplo de uso:**

```
<HospitalAdminSidebar activeTab={activeTab} setActiveTab={setActiveTab} sidebarOpen={sidebarOpen} setSidebarOpen={setSidebarOpen} />
```

- **Notas adicionales:**
  - Muestra íconos y nombres de cada sección.
  - El diseño es responsivo y moderno.

StatsCardHospital.jsx

- **Ubicación:** `src/components/hospital/StatsCardHospital.jsx`
- **¿Qué hace?** Tarjeta visual para mostrar KPIs/estadísticas de grupos o empleados a nivel hospitalario, con ícono, color y subtítulo.
- **Props principales:**
  - `icon`, `label`, `value`, `color`, `subtitle`.
- **Endpoints que consume:**
  - No consume endpoints directamente; recibe datos por props.
- **Notas sobre permisos:**
  - Solo visualización.
- **Ejemplo de uso:**

```
<StatsCardHospital icon={<Hospital />} label="Empleados" value={25} color="emerald" subtitle="Empleados registrados" />
```

- **Notas adicionales:**
  - Soporta varios colores y estilos.
  - Es reutilizable en cualquier vista hospitalaria.