**Universidad Privada del Valle**

**Facultad de Informática**



**Manual Técnico**

**Asignatura:** Proyecto de Sistemas II

**Integrantes**:

Elsa Valentina Trigo Maldonado

Angela Nicole Sánchez Leaño

Cochabamba-2025

**Manual Técnico**

**SEDES REFERENCIAS**

**Introducción:**

El presente proyecto tiene como objetivo el desarrollo de una plataforma web destinada a la gestión de información médica para el Servicio Departamental de Salud (SEDES) en Cochabamba, Bolivia. Esta plataforma tiene como propósito principal mejorar la organización y acceso a los datos relativos a hospitales, servicios médicos y especialidades dentro del departamento. Al facilitar el acceso rápido y sencillo a esta información clave, se busca mejorar la administración de los recursos de salud, optimizando la eficiencia en la toma de decisiones.

**Descripción del proyecto:**

El proyecto consiste en el desarrollo de una página web interactiva que permita al SEDES Cochabamba gestionar de manera centralizada y eficiente los registros de hospitales, servicios médicos y especialidades dentro del departamento. La plataforma permitirá registrar, modificar, eliminar y consultar información de manera intuitiva, proporcionando un sistema de fácil acceso y navegación para los usuarios.

**Funcionalidades principales:**

* **Gestión de Hospitales:** Registro, modificación y eliminación de hospitales en el sistema, con la posibilidad de visualizar una lista completa de hospitales registrados.
* **Gestión de Cartera de Servicios:** Registro, modificación y eliminación de servicios médicos ofrecidos por los hospitales, permitiendo la consulta de la lista de servicios.
* **Gestión de Especialidades:** Registro, modificación y eliminación de especialidades médicas disponibles en los hospitales, con opción para consultar una lista actualizada de especialidades.

**Link al Video demostrativo YouTube**

[**https://youtu.be/pAUjn3DRles**](https://youtu.be/pAUjn3DRles)

**Listado de Roles más sus credenciales de todos los Admin / Users del sistema:**

**Admin Sedes:** Tiene acceso total al sistema. Puede agregar nuevos hospitales, especialidades, servicios, redes de coordinación y personal de salud. Además, puede visualizar la información detallada de cada hospital, incluyendo las especialidades que ofrece, el personal de salud asignado y los turnos programados.

**Admin Hospital**: Puede gestionar las especialidades y servicios disponibles en su hospital, eligiendo cuáles estarán activos. Tiene acceso a la información del personal, la disponibilidad de camas, los turnos asignados, así como a las referencias médicas recibidas y enviadas a otros hospitales.

**Doctor**: Puede consultar las especialidades y servicios de todos los hospitales, así como las especialidades específicas del hospital al que pertenece. También tiene acceso a la información sobre la disponibilidad de camas y puede generar nuevas referencias médicas.

**Enfermera/o:** Tiene acceso únicamente a la información sobre la disponibilidad de camas en el hospital.

**Herramientas de Implementación:**

Lenguajes de programación:

JavaScript

TypeScript

Frameworks:

Frontend: React (con Vite)

Backend: NestJS (basado en Node.js)

APIs de terceros:

Librerías y paquetes npm como Axios para solicitudes HTTP.

Bases de Datos:

MySQL (gestor de base de datos relacional).

**Proceso de Dockerización del Proyecto**

**1. Creación de la Base de Datos (database/sedes\_referencia.sql)**

Se creó un archivo SQL llamado sedes\_referencia.sql, que contiene el script de inicialización de la base de datos sedes\_referencia.

Este archivo se usa para crear las tablas y poblar datos iniciales automáticamente cuando se levanta el contenedor de MySQL.

**2. Configuración de docker-compose.yml**

Se creó un archivo docker-compose.yml para definir y levantar todos los servicios de manera conjunta. Se usó la versión 3.8 de Compose.

Los servicios definidos fueron:

Servicio mysql

Imagen: mysql:8.0, utilizando una versión estable de MySQL.

Contenedor: Nombrado como mysql\_sedes.

Variables de entorno: Se definió MYSQL\_ROOT\_PASSWORD para la contraseña del root y MYSQL\_DATABASE para crear automáticamente la base de datos sedes\_referencia.

Puertos: Se mapeó el puerto 3306 interno de MySQL al 3307 del host, para evitar conflictos con otros MySQL locales.

Volumen: Se montó sedes\_referencia.sql dentro de /docker-entrypoint-initdb.d/, para que al iniciar el contenedor, MySQL ejecute este script y cree la base de datos automáticamente.

Red: Se conectó a una red virtual llamada app-network para comunicación interna entre servicios.

Servicio backend

Build: Se construyó desde el directorio ./backend-nest-dev, donde está el proyecto NestJS.

Contenedor: Nombrado como backend\_sedes.

Variables de entorno: Se especificó el archivo .env para configurar variables como conexión a la base de datos.

Puertos: Se expuso el puerto 3000, que es donde el backend NestJS corre por defecto.

Dependencias: Se utilizó depends\_on para indicar que el backend debe esperar que mysql esté disponible primero.

EntryPoint: Se configuró el script wait-for-it.sh para esperar a que MySQL esté listo antes de iniciar el servidor NestJS, evitando errores de conexión por arranque apresurado.

Red: Se conectó también a app-network.

Servicio frontend

Build: Se construyó a partir de ./Frontend-React-Dev, donde se encuentra el frontend hecho en React.

Contenedor: Nombrado como frontend\_sedes.

Variables de entorno: Se cargó el archivo .env para configuración del frontend.

Puertos: Se expuso el puerto 5173, utilizado por Vite para desarrollo de React.

Dependencias: Se configuró depends\_on para que espere que el backend esté arriba antes de iniciar.

Command: Se corrió el comando npm run dev -- --host, permitiendo que el servidor Vite sea accesible desde fuera del contenedor.

Red: Conectado a app-network igual que los otros servicios.

Red (app-network)

Se creó una red tipo bridge para permitir que los contenedores se comuniquen internamente por nombre.

**3. Dockerfile del Backend (backend-nest-dev/Dockerfile)**

Imagen base: Se usó node:18, una versión estable de Node.js.

Directorio de trabajo: /app.

Instalación de dependencias: Se copiaron los archivos package.json y package-lock.json y luego se corrió npm install.

Copia de código fuente: Se copió todo el proyecto al contenedor.

Comando por defecto: npm run start:dev, que inicia NestJS en modo de desarrollo.

**4. Script de Espera (backend-nest-dev/wait-for-it.sh)**

Se incluyó el script wait-for-it.sh, que:

Espera que el servicio MySQL esté disponible antes de iniciar el backend.

Reintenta conexiones hasta que se pueda establecer una conexión al puerto 3306.

Permite evitar errores como "ECONNREFUSED" al iniciar el backend mientras MySQL aún no ha terminado de arrancar.

**5. Dockerfile del Frontend (Frontend-React-Dev/Dockerfile)**

Imagen base: También se usó node:18.

Directorio de trabajo: /app.

Instalación de dependencias: Se copiaron package.json y package-lock.json para ejecutar npm install.

Copia de código fuente: Se copió todo el proyecto React al contenedor.

Exposición de puerto: Se expuso el puerto 5173, usado por Vite.

Comando por defecto: npm run dev, que levanta el servidor de desarrollo de Vite.

**Instalación y configuración:**

Instrucciones detalladas sobre cómo instalar el software.

1. Clonar el repositorio del proyecto:

git clone <https://github.com/Univalle-PSII/sedes-referencias.git>

2. Abrir Docker  
 3. En la carpeta del proyecto abrir el símbolo de sistema (cmd) y pegar los siguientes códigos:

docker-compose build --no-cache

docker-compose up

4. Ir a <http://localhost:5173/inicio>

**Credenciales de Acceso al Sistema**

Roles y usuarios

1. Rol: Sedes de administrador

Usuario: jlopezt

Contraseña: admin1234