2025

Documentacion del Front – Sistema de gestión hospitalaria

ANGEL DAVID REYES TELLEZ
LUIS IVAN MARQUEZ AZUARA
BRAYN KALID REYES SILVA
ALDO TOLENTINO DOMINGUEZ
IRVING MORALES

UNIVERSIDAD TECNOLOGICA DE XICOTEPEC DE JUAREZ |

Indice

Introduccion

El frontend del Sistema de Gestión Hospitalaria es una interfaz moderna desarrollada para optimizar la administración de recursos médicos. Diseñada con Vue.js 3 y Bootstrap 5, ofrece un flujo de trabajo intuitivo para gestionar pacientes, personal médico e inventarios.

Objetivos clave:

- Centralizar información clínica y operativa.
- Reducir errores humanos mediante validaciones en tiempo real.
- Garantizar accesibilidad en dispositivos móviles y desktop.

La plataforma integra tecnologías como Socket.IO para notificaciones instantáneas y Chart.js para visualización de datos estadísticos, cumpliendo con los estándares de usabilidad y rendimiento.

Tecnologias Utilizadas

Tecnología	Versión	Descripción
Vue.js	3.5.13	Framework progresivo para construir interfaces de usuario. Permite componentes reactivos y escalabilidad.
Bootstrap	5.1.3	Biblioteca de estilos para diseño responsivo y componentes UI predefinidos (modales, formularios).
Axios	1.8.2	Cliente HTTP para conexiones con la API backend. Gestiona solicitudes asíncronas y manejo de errores.
Socket.IO	4.8.1	Biblioteca para comunicación en tiempo real (ej: alertas de emergencia o actualizaciones de inventario).
Chart.js	4.4.8	Generación de gráficos interactivos (ej: reportes de ocupación hospitalaria).
Swiper	11.2.5	Slider interactivo para galerías de imágenes (ej: tutoriales de uso).
Vue Router	4.5.0	Sistema de navegación entre vistas (ej: Login → Dashboard).

Wireframe y prototipo

Los **wireframes en baja fidelidad** (elaborados a mano y posteriormente los prototipos en figma) establecieron los cimientos de la arquitectura de información del sistema, priorizando:

1. **Jerarquía visual**: Distribución clara de elementos críticos (ej: barra de navegación siempre visible, ubicación estratégica de botones de acción primaria).
2. **Flujo de usuario**: Secuencia lógica entre pantallas (ej: Login → Dashboard → Módulo Pacientes), reduciendo pasos innecesarios.

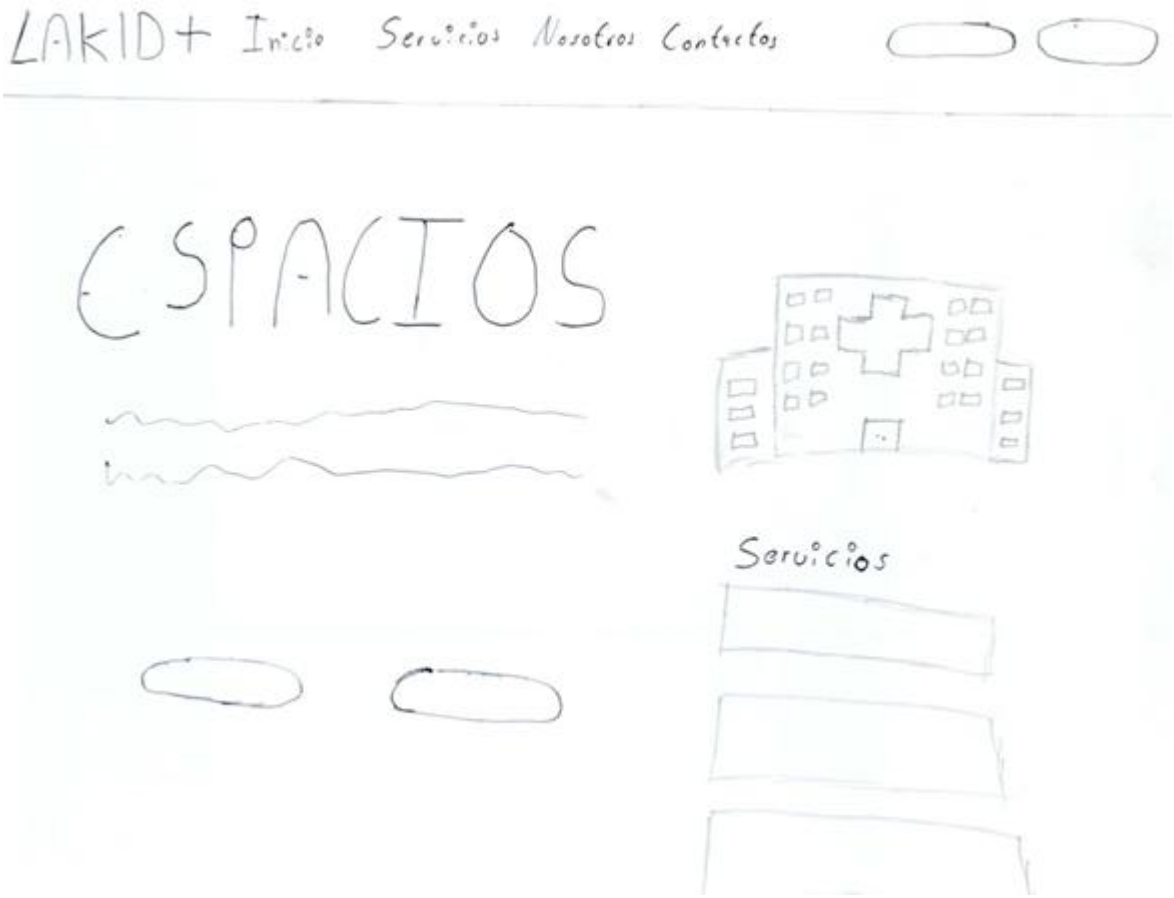

Imagen 1. Wireframe de la landing page de LAKID Systems

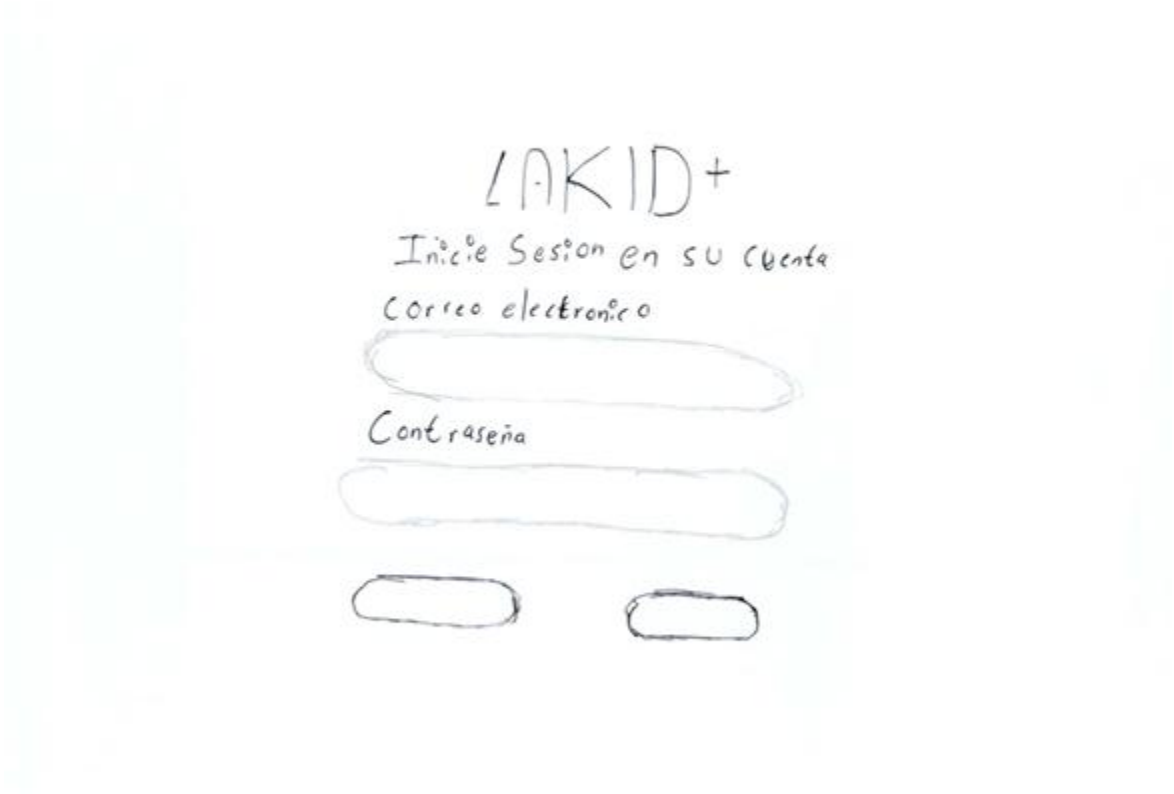

Imagen 2. Wireframe de la pagina de login de LAKID systems

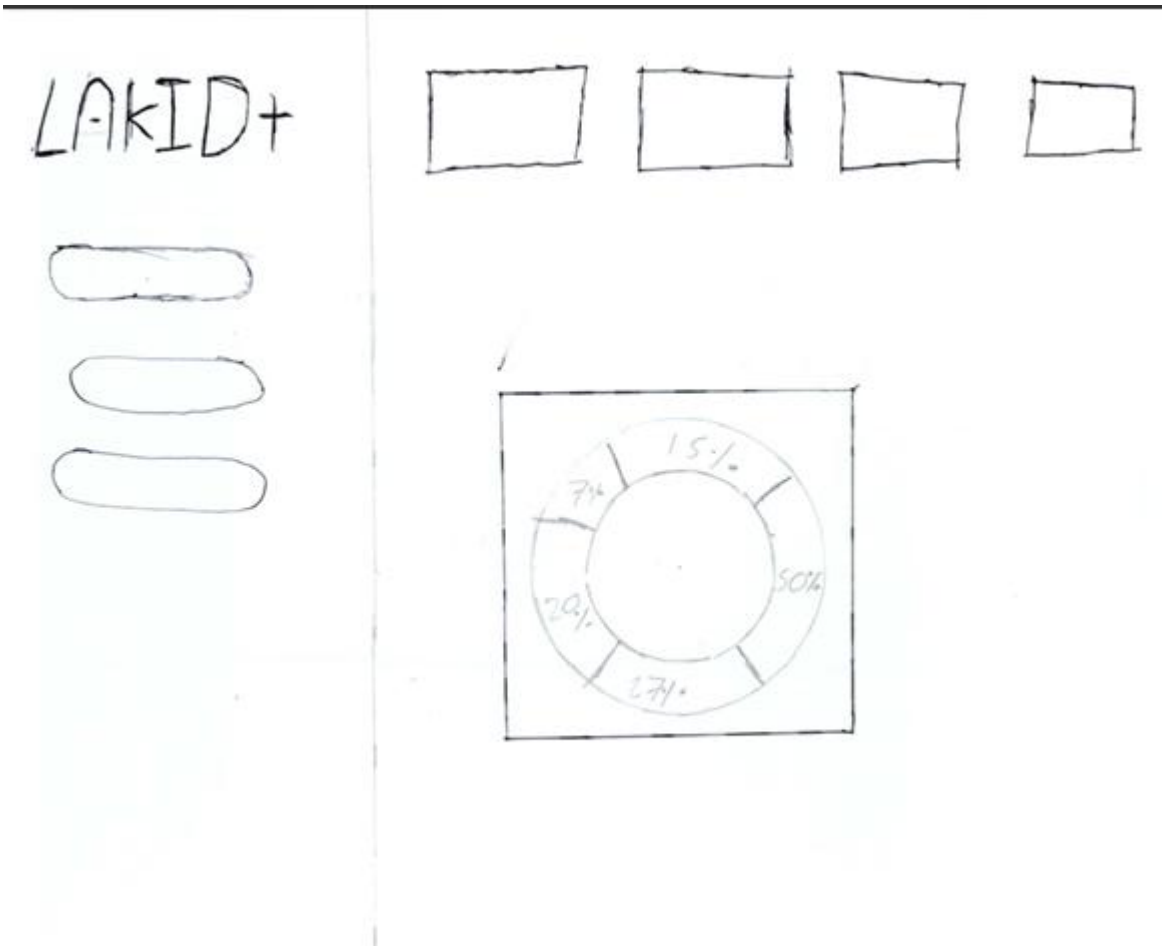

Imagen 3. WireFrame de la pagina de Dashboard

Prototipo

El prototipo realizado en figma muestra una versión mas detallada del proyecto, esto con la finalidad de dar una visión mas fiel al producto final, cuenta con desplazamiento entre pantallas, paleta de clores y mas detalles solicitados en las diferentes revisiones.

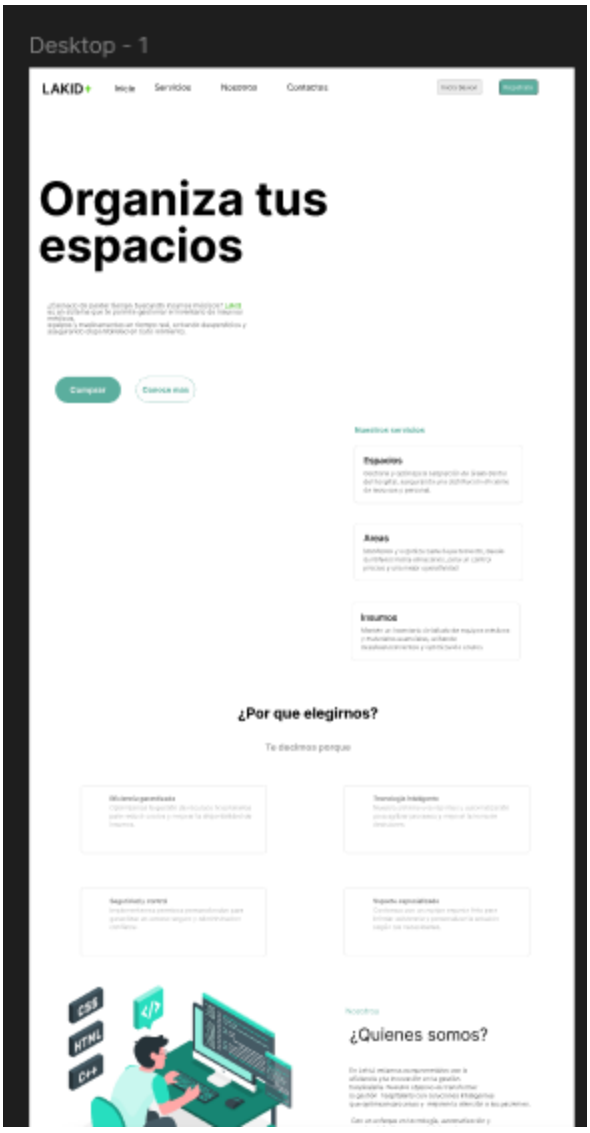

Imagen 4. Prototipo de la landing page en figma

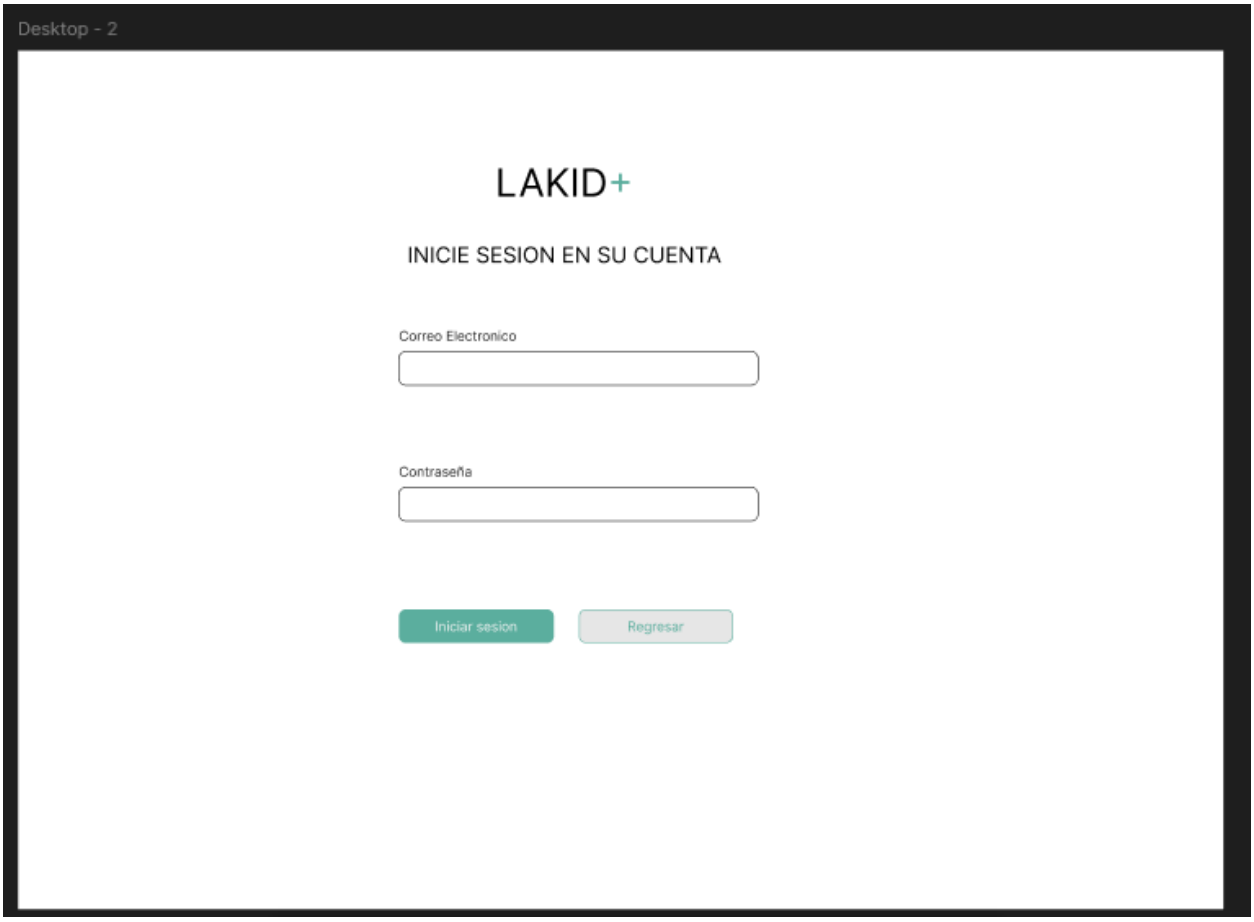

Imagen 6. Prototipo del login en figma

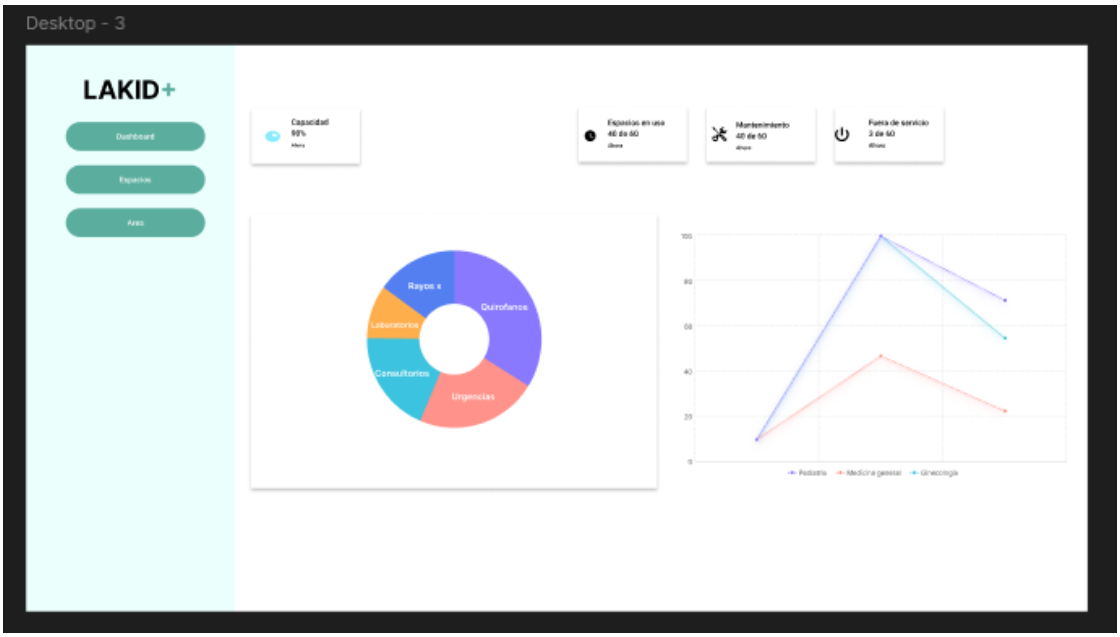

Imagen 7. Prototipo del Dashboard en figma

Estructura del proyecto

PROYECTO	PUBLIC	CSS Contiene archivos de estilo
		FONTS Contiene las fuentes de la aplicacion
		IMAGES Contiene las imagenes necesarias
		JS *Scripts externos o librerías de JavaScript que no se gestionan por npm (ej: plugins legacy).*
	SRC	ASSETS Recursos estaticos internos
		COMPONENTS Componentes reutilizables
		PAGES Vistas principales de la aplicacion
		ROUTER Configuracion de rutas con Vue Router
		APP.VUE Componente Raiz de la aplicacion
		MAIN.JS Punto de entrada de la aplicacion

Componentes clave

Espacios.Vue

1. Propósito

Componente que muestra y gestiona espacios hospitalarios (quirófanos, salas, etc.) mediante:

- Tabla interactiva con filtrado automático.
- Formulario modal para agregar/editar espacios.
- Actualización en tiempo real (cada 3 segundos).

---

2. Estructura y Funcionalidades

Sección	Descripción
Tabla Principal	Muestra tipo, nombre, estatus (coloreado) y capacidad de cada espacio.
Botón "Agregar"	Abre modal para registrar nuevos espacios.
Modal de Formulario	Campos: tipo, nombre, estatus (select), capacidad (numérico).
Auto-actualización	Consulta API cada 3 segundos (setInterval).

---

3. Props y Data

Variable/Prop	Tipo	Descripción
rooms	Array	Lista de espacios ({ tipo, nombre, estatus, capacidad }).
mostrarLogin	Boolean	Controla visibilidad del modal (true/false).
nuevoEspacio	Object	Datos del nuevo espacio (bind con formulario).

---

4. Métodos Clave

Método	Descripción
getEspacios()	Fetch GET a https://integradora-backend-linux.onrender.com/espacios. Maneja token JWT.
agregarEspacio()	Envía POST con nuevos datos a la API. Valida campos y genera UUID automático.
startAutoUpdate()	Inicia intervalo de actualización automática (3 segundos).
generateUUID()	Genera ID único para nuevos registros.

---

5. Estilos Destacados

- **Coloración de estatus**:
    - activo: Verde (#3b8e6c).
    - ocupado: Rojo (#c97e6b).
    - mantenimiento: Azul (#6ea87d).

Grafica_area_espacios.vue

(Componente de gráfica interactiva de uso de espacios médicos)

1. Propósito

Visualiza en tiempo real los **5 espacios médicos más utilizados** en formato de gráfico polar (PolarArea), con:

- Auto-actualización cada 3 segundos.
- Colores diferenciados por espacio.
- Diseño responsive (adaptable a pantallas).

---

2. Tecnologías Clave

Tecnología	Uso en el Componente
Chart.js	Gráfico polar interactivo.
Fetch API	Conexión con API externa (space-api-7odx.onrender.com).
Vue 3	Composition API (<script setup>).

---

3. Funcionalidades Principales

Método/Función	Descripción
obtenerDatos()	Fetch GET a la API para obtener datos crudos de espacios.
contarFrecuenciaEspacios()	Procesa datos para contar usos por espacio (top 5).
actualizarGrafico()	Renderiza/actualiza el gráfico con Chart.js (polarArea).
iniciarActualizacionPeriodica()	Ejecuta cargarDatos() cada 3 segundos (setInterval).

---

4. Data y Lógica Clave

Variable	Tipo	Descripción
chartInstance	Object	Instancia del gráfico Chart.js.
lastData	Array	Almacena últimos datos para evitar renders innecesarios.
interval	Number	ID del intervalo de auto-actualización.

6. Estilos Destacados

- **Gráfico**: Colores pastel (rgba(91, 174, 158, 0.6) para el espacio más usado)

Header.vue

1. Propósito

Barra de navegación superior con:

- Enlaces estáticos (Inicio, Servicios, Nosotros, Contactos).
- Botones dinámicos (**Iniciar Sesión** y **Registrarse**) con estilos personalizados.
- Diseño **responsive** (menú hamburguesa en móviles).
- Integración con **Vue Router** para navegación SPA.

2. Tecnologías Clave

Tecnología	Uso en el Componente
Vue 3	<script setup> + Reactividad.
Bootstrap	Estructura responsive (opcional, clases similares).
Vue Router	Enlace a /login con <router-link>.
CSS Personalizado	Estilos para botones (cta-primary, cta-button-outline).

3. Estructura y Funcionalidades

Elemento	Descripción
Logo	Muestra siteName + ícono médico (➕) en color #5bae9e.
Enlaces	Secciones fijas (#home-section, #services-section, etc.).
Botones	- Inicia Sesión: Redirige a /login (Vue Router). - Regístrate: Botón destacado (themeColor dinámico).
Menú Móvil	Botón hamburguesa (.js-menu-toggle) que colapsa/expande en pantallas pequeñas.

4. Data y Personalización

Variable	Origen	Uso
siteName	../data/items.js	Nombre del sitio (ej: "Hospital XYZ").
themeColor	../data/items.js	Color del botón "Regístrate" (ej: #5bae9e).

Instalación y Configuración del Frontend

1. Requisitos Previos

Antes de ejecutar el proyecto, asegúrate de tener instalado:

- **Node.js** (v16 o superior): Entorno de ejecución para JavaScript.
    - Descarga: https://nodejs.org/
- **npm** (viene con Node.js): Gestor de paquetes para instalar dependencias.
- **Git** (opcional): Para clonar el repositorio.

---

2. Pasos para Configurar el Proyecto

1. Instalar Dependencias

El proyecto usa librerías como **Vue.js, Axios y Bootstrap**. Para instalarlas:

npm install

¿Qué hace este comando?

- Lee el archivo package.json y descarga todas las dependencias listadas en dependencies y devDependencies.
- Crea la carpeta node_modules (no se sube al repositorio).

3. Configurar Variables de Entorno

El frontend necesita conectarse a la API. Crea un archivo .env en la raíz del proyecto con:

Un archivo .env

4. Ejecutar el Servidor de Desarrollo

Para levantar el frontend en modo desarrollo:

npm run dev

¿Qué ocurre?

- Se inicia un servidor local (generalmente en http://localhost:8080).
- Vue.js recarga automáticamente los cambios (Hot Reload).

---

3. Posibles Errores y Soluciones

Error	Causa	Solución
Error: Cannot find module	Dependencias no instaladas.	Ejecutar npm install nuevamente.
CORS bloqueado	La API no permite conexiones desde el origen del frontend.	Verificar URL en .env o configurar CORS en el backend.
Puerto en uso	Otro programa usa el puerto 8080.	Cambiar el puerto en vite.config.js

4. Estructura de Archivos Clave

```
RM/
├── node_modules/  # Dependencias instaladas (npm install)
├── public/        # Assets estáticos (imágenes, favicon)
├── src/
|   ├── components/ # Componentes Vue (Navbar.vue, Espacios.vue)
|   ├── App.vue     # Componente principal
|   └── main.js     # Punto de entrada de la app
├── .env           # Variables de entorno
└── package.json   # Lista de dependencias y scripts
```

5. Recomendaciones

- **Primera ejecución**: Verifica que no haya errores en la terminal después de npm run dev.
- **Dependencias adicionales**: Si añades más librerías (ej: Chart.js), instálalas con:

```
npm install chart.js
```