Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

**Materia**Desarrollo de Sistemas Geo-Referenciados

**Carrera**Ingeniería de Software y Sistemas Computacionales  
(ISSC - 612)

**Alumno**Diego Camarena Gutiérrez

**Entrega**Proyecto parcial documentación técnica

18/03/2025

Contenido

[Documentación Técnica 3](#_Toc193232050)

[Descripción general 3](#_Toc193232051)

[Tecnologías utilizadas 3](#_Toc193232052)

[Front 3](#_Toc193232053)

[API’s y herramientas 3](#_Toc193232054)

[Estructura del proyecto 3](#_Toc193232055)

[Instalación y configuración 4](#_Toc193232056)

[Funcionamiento 4](#_Toc193232057)

[index.html 4](#_Toc193232058)

[styles.css 4](#_Toc193232059)

[script.js 4](#_Toc193232060)

[Referencias 9](#_Toc193232061)

# Documentación Técnica

## Descripción general

Esta es una herramienta que hace un análisis sobre la cobertura hospitalaria sobre la ubicación que el usuario ingrese mediante clic o buscando la dirección.

Se utiliza OpenStreetMap y Overpass API, para la extracción de datos y el análisis de las ubicaciones.

## Tecnologías utilizadas

### Front

* HTML5: Estructura de la aplicación.
* CSS3: Estilos para mejorar la apariencia de la app.
* JavaScript: Lógica de la app y manipulación del DOM.
* Leaflet.js: Librería para el uso de mapas interactivos.

### API’s y herramientas

* OpenStreetMap (OSM): Proveedor de mapas base.
* Overpass API: Consulta de hospitales en un área geográfica definida.
* Nominación Geocoding API: Conversión de direcciones en coordenadas geográficas.

## Estructura del proyecto

Texto

El contenido generado por IA puede ser incorrecto.

## Instalación y configuración

* Para poder utilizar este programa se necesita descargar la carpeta completa de PROYECTO PARCIAL 1, en la cual se encuentran los archivos necesarios para poder ejecutar la app web.
* Repo <https://github.com/Diego-CGTZ/Cobertura-de-hospitales.git>
* Se necesita contar con un navegador moderno y actualizado como Chrome, Edge, Brave, etc.
* El siguiente paso es abrir el archivo index.html

## Funcionamiento

index.htmlEste archivo contiene la estructura del programa, contiene la interfaz de usuario, vincula el script con el css y también incluye la librería de Leaflet para el uso de mapas.

styles.cssEste archivo define el diseño de la aplicación, personalizando el panel de control y el mapa.

script.jsEn este archivo se concentra toda la lógica del programa, la cual se desglosa a continuación.

**Declaración de variables**

let map, markers = [];

const RANGES = {

    cercanos: 1000,    // 1 km

    medios: 3000,     // 3 km

    lejanos: 5000     // 5 km

};

Se declaran **map** y **markers** como listas vacías, en map se va a almacenar el objeto del mapa y la lista de hospitales para mostrar en el mapa, también se declara **RANGES** donde se declaran los estándares para evaluar que se considera tener hospitales cercanos, medios y lejos.

**Inicialización del mapa**

async function initMap() {

    map = L.map('map').setView([40.4168, -3.7038], 13);

    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

        attribution: '&copy; OpenStreetMap contributors'

    }).addTo(map);

    map.on('click', async e => {

        await analyzeLocation(e.latlng);

    });

}

En este caso se declara la función asíncrona para poder inicializar el mapa, lo que implica que esta función se pueda ejecutar en “segundo plano” mientras se continua ejecutando este programa.

Se crea el **mapa** usando Leaflet.js, centrado en Madrid en este caso.

Se añade la **capa base** de OpenStreetMap y también al hacer clic sobre el mapa se ejecuta analizeLocation para dar análisis sobre esa zona.

**Geocodificación de Direcciones**

async function geocodeAddress() {

    const address = document.getElementById('address').value;

    const response = await fetch(`https://nominatim.openstreetmap.org/search?format=json&q=${address}`);

    const data = await response.json();

    if (data.length > 0) {

        const location = L.latLng(data[0].lat, data[0].lon);

        map.flyTo(location, 15);

        await analyzeLocation(location);

    }

}

En esta función se **obtienen las coordenadas geográficas** a partir de la dirección ingresada por el usuario, se usa la **API de Nominatim** para hacer la conversión, también se encarga de **mover el mapa** a la ubicación encontrada y **analiza la cobertura** con la función **analyzeLocation**.

**Análisis de cobertura hospitalaria**

async function analyzeLocation(location) {

    markers.forEach(marker => map.removeLayer(marker));

    markers = [];

    const overpassQuery = `

        [out:json];

        node[amenity=hospital](around:${RANGES.lejanos},${location.lat},${location.lng});

        out;

    `;

    try {

        const response = await fetch(`https://overpass-api.de/api/interpreter?data=${encodeURIComponent(overpassQuery)}`);

        const data = await response.json();

        let closest = { distance: Infinity };

        const counts = { cercanos: 0, medios: 0, lejanos: 0 };

        data.elements.forEach(hospital => {

            const hospitalLoc = L.latLng(hospital.lat, hospital.lon);

            const distance = location.distanceTo(hospitalLoc);

            if (distance <= RANGES.cercanos) counts.cercanos++;

            else if (distance <= RANGES.medios) counts.medios++;

            else if (distance <= RANGES.lejanos) counts.lejanos++;

            if (distance < closest.distance) {

                closest = {

                    distance,

                    name: hospital.tags.name || 'Hospital sin nombre',

                    coords: hospitalLoc

                };

            }

            const marker = L.marker(hospitalLoc, {

                icon: L.divIcon({ className: 'hospital-marker', iconSize: [12, 12] })

            }).bindPopup(`${hospital.tags.name}<br>${(distance / 1000).toFixed(2)} km`);

            markers.push(marker.addTo(map));

        });

        const total = counts.cercanos + counts.medios + counts.lejanos;

        let coverageLevel = 'low';

        if (total === 0) {

            coverageLevel = 'low';

        } else {

            const score = (counts.cercanos \* 3) + (counts.medios \* 2) + counts.lejanos;

            if (score >= 5) coverageLevel = 'good';

            else if (score >= 2) coverageLevel = 'moderate';

        }

        const resultDiv = document.getElementById('analysisResult');

        resultDiv.innerHTML = `

            <h3 style="margin: 5px 0;">Análisis de cobertura:</h3>

            ${total === 0 ?

                '<div class="coverage-low">❌ Sin hospitales cercanos</div>' :

                `<div class="coverage-${coverageLevel}">

                    <strong>Nivel de cobertura:</strong> ${getCoverageLabel(coverageLevel)}

                </div>

                <div style="margin-top: 10px;">

                    <strong>Hospital más cercano:</strong><br>

                    ${closest.name}<br>

                    ${(closest.distance / 1000).toFixed(2)} km

                </div>

                <div style="margin-top: 10px;">

                    <strong>Distribución:</strong><br>

                    • <1 km: ${counts.cercanos}<br>

                    • 1-3 km: ${counts.medios}<br>

                    • 3-5 km: ${counts.lejanos}<br>

                    Total: ${total}

                </div>`

            }

        `;

    } catch (error) {

        console.error('Error:', error);

    }

}

Esta función **elimina** los marcadores marcados anteriormente en el mapa, **construye** la consulta a la API para encontrar los hospitales dentro del rango máximo, **clasifica** hospitales según su distancia, **identifica** el hospital más cercano y **añade** marcadores de hospitales en el mapa.

**Cálculo del nivel de cobertura**

function getCoverageLabel(level) {

    const labels = {

        good: '🟢 Buena cobertura',

        moderate: '🟡 Cobertura moderada',

        low: '🔴 Cobertura insuficiente'

    };

    return labels[level] || 'Desconocida';

}

Esta función **define** el texto y colores para los niveles de cobertura.

## Referencias

*Overpass API - OpenStreetMap Wiki*. (2022). Openstreetmap.org. <https://wiki.openstreetmap.org/wiki/Overpass_API>

*Descripción general de la API de Geocoding*. (2025). Google for Developers. <https://developers.google.com/maps/documentation/geocoding/overview?hl=es-419>