|  |
| --- |
|  |

MANUAL TÉCNICO

PROTOTIPO DEL GEOPORTAL

UNIVERSIDAD CATÓLICA DE CUENCA

CUENCA - ECUADOR

Revidado por: Junior Wachapa

Responsable: Sandra Cobos

Publicado por: SIGDATA, 2024

**Tabla de Contenido**

[INTRODUCCIÓN 3](#_Toc19989)

[OBJETIVOS 4](#_Toc19990)

[1. REQUERIMIENTOS TÉCNICOS 5](#_Toc19991)

[2. HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO DEL GEOPORTAL 6](#_Toc19992)

[Visual Studio Code 6](#_Toc19993)

[Geoserver 6](#_Toc19994)

[GitHub 6](#_Toc19995)

[3. ARQUITECTURA DEL SISTEMA 7](#_Toc19996)

[4. COMPONENTES DEL SOFTWARE 9](#_Toc19997)

[Catálogo de Proyectos 9](#_Toc19998)

[Geoservicios 11](#_Toc19999)

[Manual 16](#_Toc20000)

[Geovisor 20](#_Toc20001)

[Contactos 23](#_Toc20002)

# INTRODUCCIÓN

En este manual describe los pasos necesarios para cualquier persona que tengan ciertas bases de sistemas para que pueda realizar los cambios, la edición de todo el prototipo.

En este manual se mencionan las especificaciones mínimas de hardware y software para el correcto uso del prototipo. Y describe las herramientas clave utilizadas en el desarrollo, como Visual Studio Code, GeoServer, y GitHub. Explica brevemente la arquitectura del sistema y detalla las tecnologías y frameworks implementados, incluyendo Node.js, React, y Tailwind CSS, entre otros. Además, proporciona una visión general de los componentes principales del software, como el catálogo de proyectos, los geoservicios, y el geovisor, enfocándose en su propósito y contribución al proyecto. Este manual sirve como una referencia esencial para realizar cambios y ediciones en el prototipo, asegurando su correcta utilización.

# OBJETIVOS

* Brindar la información necesaria para poder realizar la instalación y configuración del prototipo.
* Describir las herramientas utilizadas para el diseño y desarrollo del prototipo.
* Brindar instrucciones paso a paso y orientación práctica que permitan a los usuarios realizar cambios, ediciones y personalizaciones en el prototipo, promoviendo la adaptabilidad y mejora continua del sistema.

# 1. REQUERIMIENTOS TÉCNICOS

**REQUERIMIENTOS MÍNIMOS DE HARDWARE**

* **Procesador:** 1.6 GHz or faster processor
* **Memoria RAM:** 1 GB of RAM • **Memoria Interna:** 240Gb

### REQUERIMIENTOS MÍNIMOS DE SOFTWARE

• **Sistema Operativo:** Windows 10 and 11 (64-bit) macOS versions con Apple security update support. Linux (Debian): Ubuntu Desktop 20.04, Debian 10

Linux (Red Hat): Red Hat Enterprise Linux 8, Fedora 36

# 2. HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO DEL GEOPORTAL

# Visual Studio Code

Microsoft lo desarrolló, un editor avanzado de código fuente distribuido de forma gratuita y compatible con sistemas operativos como Windows, Linux y macOS. Este editor se caracteriza por su capacidad para facilitar el proceso de desarrollo de software, gracias a características como la depuración integrada, el manejo eficiente de versiones con Git, el resaltado de sintaxis, y el autocompletado de código, conocido también como IntelliSense. Además, ofrece personalizar la interfaz de usuario, los atajos de teclado y las configuraciones de trabajo mediante un amplio catálogo de extensiones disponibles en su tienda en línea, adaptando el entorno de desarrollo a muchos lenguajes de programación y marcos de trabajo. Visual Studio Code es una herramienta indispensable para los desarrolladores que buscan optimizar su flujo de trabajo, desde la edición de simples fragmentos de código hasta la gestión de proyectos complejos, promoviendo la colaboración y la eficiencia en equipos de desarrollo.

# Geoserver

GeoServer es un servidor de software libre enfocado en la compartición y edición de datos geoespaciales, promoviendo la interoperabilidad mediante la adhesión a los estándares abiertos del Open Geospatial

Consortium (OGC), como Web Feature Service (WFS), Web Map Service (WMS), y Web Coverage Service (WCS). Esta plataforma facilita el acceso y análisis de información geográfica a través de internet, permitiendo a los usuarios visualizar mapas y conectar con diversas fuentes de datos geoespaciales, incluyendo bases de datos PostGIS, archivos shapefile y mosaicos de imágenes. GeoServer es una herramienta esencial en el ámbito de los Sistemas de Información Geográfica (SIG) para el desarrollo de aplicaciones web GIS, publicación de datos espaciales en la red y la integración de estos datos en diferentes plataformas y servicios, sirviendo como solución versátil para organizaciones que desean compartir información geográfica ampliamente o facilitar la colaboración en equipos de trabajo.

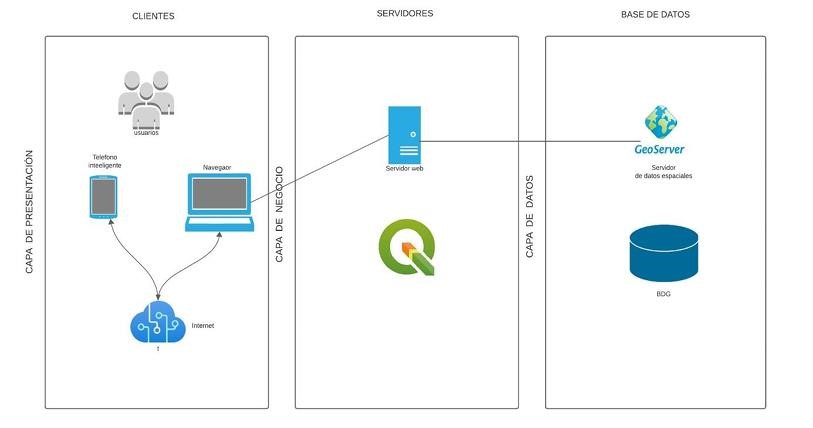
# GitHub

Se trata de una plataforma de desarrollo colaborativo diseñada para alojar proyectos de software, aprovechando el sistema de control de versiones Git. Esta herramienta es fundamental para gestionar repositorios de código, ofreciendo diversas funcionalidades que facilitan el trabajo en equipo dentro de un proyecto. Uno de los usos más frecuentes es mantener el repositorio actualizado con las contribuciones de cada miembro del equipo, para lo cual se emplean comandos como git commit, git merge, git pull y git push, entre otros, que permiten integrar y compartir el trabajo realizado de manera eficiente.

Para la IDE-UCACUE, se gestiona un repositorio privado al que se le aportan diversos commits con ajustes o modificaciones efectuadas en el software, facilitando así la colaboración y el trabajo en equipo en el desarrollo de la aplicación.

# 3. ARQUITECTURA DEL SISTEMA

**Diagramas de Arquitectura: Incluye diagramas que ilustran la estructura del sistema.**



**Tecnologías Utilizadas: Detalla las tecnologías, frameworks y herramientas implementadas.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Herramienta** | **¿Qué es?** | **Uso** | **Importancia** | **Función**  **Principal** | **Recurso** |
| Node.js | Entorno de ejecución para JS | Desarrollo de aplicaciones web  y servidores | Fundamental en el desarrollo del lado del servidor | Permite ejecutar    código JavaScript    fuera de[l](https://nodejs.org/en)  navegador | <https://nodejs.org/en> |
| React | Biblioteca de  JavaScript | Construcción de interfaces de usuario interactivas | Dominante en el desarrollo de  interfaces web | Facilita la creación    de componentes    reutilizables e[n](https://es.react.dev/)  aplicaciones web | <https://es.react.dev/> |
| Tailwind  CSS | Framework de estilos en  CSS | Estilización  rápida y    consistente de  interfaces | Simplifica la gestión y personalización  del estilo | Proporciona clases utilitarias para estilos predefinidos | <https://tailwindcss.com/> |
| Leaflet | - Es una biblioteca de  JavaScript.  - Se utiliza para crear mapas  interactivos en la web. | Permite a los  usuarios  interactuar con los mapas, como hacer zoom, arrastrar, y agregar marcadores | Proporciona una  solución sencilla y eficiente para la integración de  mapas. | Permite personalizar mapas con varias capas, como imágene[s](https://leafletjs.com/)  satelitales, rutas de transporte, y otros puntos de interés. | <https://leafletjs.com/> |
| Nodemailer | Librería de Node.js para enviar emails. | Enviar emails    desde    aplicaciones    Node.js. | Notificar a usuarios vía email  (registros, contraseñas, notificaciones). | Facilitar envío de emails vía SMTP [o](https://www.nodemailer.com/) servicios de correo. | <https://www.nodemailer.com/> |
|  |

# 4. COMPONENTES DEL SOFTWARE

Descripción de Componentes: Enumera y describe cada componente del software y su propósito.

# Catálogo de Proyectos

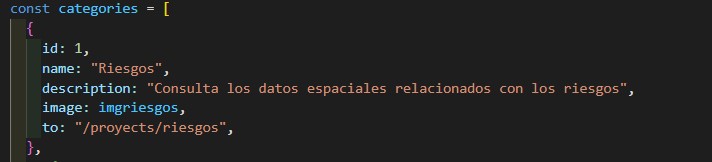
En este componente se muestra lo que son la categoría de proyectos, publicaciones destacadas, los filtros y un ejemplo de proyecto que pertenece a esa categoría.

### Categoría de Proyectos

Este fragmento de código está dentro de la clase “**categories.jsx**” la cual esta parte define una constante “**categories**” la cual es un arreglo que contiene objetos. Cada objeto representa una categoría de información o funcionalidad dentro de una aplicación, que está relacionada con información geográfica o servicios de mapas.

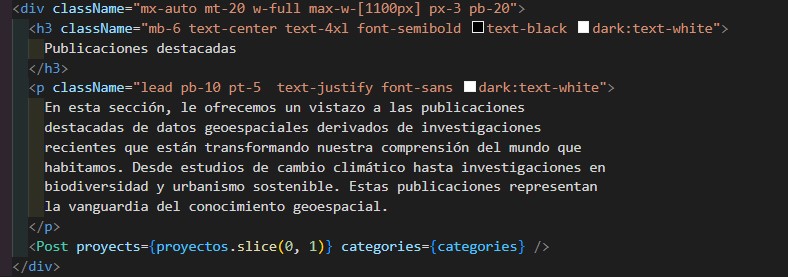
### Documentación Específica

* **Propósito del Arreglo “categories”:** Explicar que este arreglo tiene como objetivo agrupar y organizar distintas categorías de información o funcionalidades que ofrece la aplicación. Es especialmente útil para la construcción dinámica de menús, navegación, o presentaciones de categorías en la interfaz de usuario.
* **Usos de la Interfaz de Usuario:** Sugerir formas en las que este arreglo podría ser utilizado en la interfaz de usuario, como la generación de una lista o cuadrícula de categorías, incluyendo la presentación de la imagen, el nombre, y la descripción de cada categoría.



### Publicaciones Destacadas

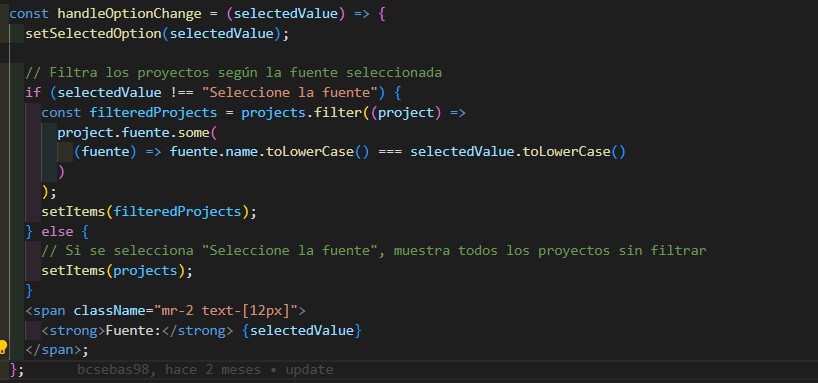
Este fragmento de código está dentro de la clase “**CatalogoPage.jsx**” es donde se muestra de que se trata las publicaciones destacadas relacionadas con datos geoespaciales, y luego de eso está la línea de código que es el componente “**Post**” donde es instanciado, pasando por un arreglo que contiene el primer elemento de “**proyectos**” y arreglo completo “**categories**”. Esto permite que el componente “**Post**” tenga acceso a esos datos para su funcionamiento interno, como la renderización de información sobre el primer proyecto y categorizarlo o filtrarlo basado en las “**categories**” proporcionadas.



### Filtros

Este fragmento de código está dentro de la clase “**ProyectPage.jsx**” tiene la función para manejar la selección de una opción por parte del usuario, filtrar un conjunto de datos basándose en esta selección y actualizar el estado de la aplicación para reflejar los resultados filtrados.

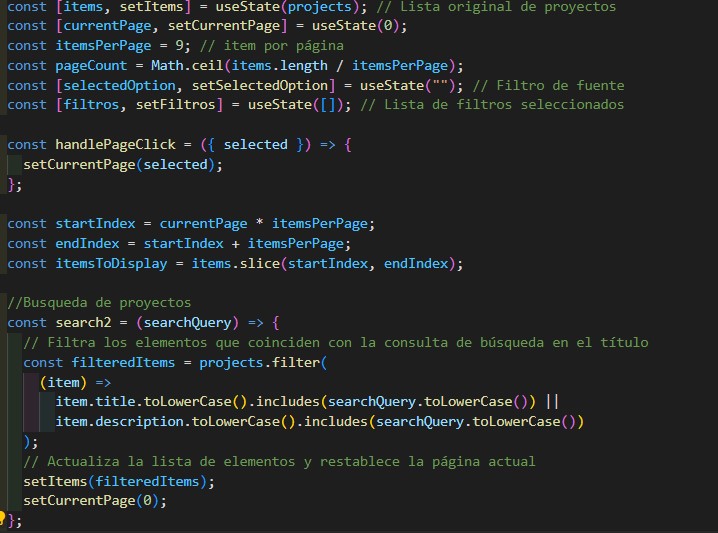
• **Filtrado de Proyectos:** La función primero verifica si el valor seleccionado es diferente de "**Seleccione la fuente**". Si es así, procede a filtrar el arreglo “**projects**”, buscando proyectos cuya propiedad “**fuente**” contenga al menos una coincidencia con el valor seleccionado. La comparación se realiza de manera insensible a mayúsculas y minúsculas.



### Búsqueda de Proyectos

Este fragmento de código que utiliza hooks para gestionar el estado y la lógica de visualización de una lista paginada de proyectos, junto con capacidades de filtrado y búsqueda.

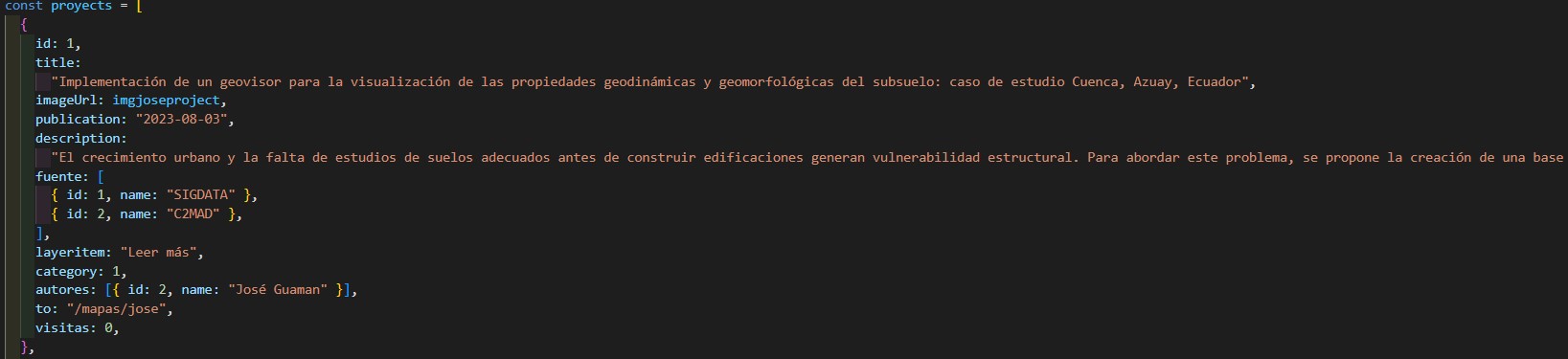
• **Documentación Específica:** Describir cómo los hooks de estado se utilizan para manejar la interactividad de la lista de proyectos, incluyendo la paginación, el filtrado y la búsqueda.



### Ejemplo de Proyecto

Este fragmento de código está dentro de la clase “**proyects.jsx**” que contiene objetos, cada uno representando un proyecto específico. Este arreglo está orientado a ser utilizado en un contexto donde se manejan datos geoespaciales o proyectos relacionados con la geografía, como se indica por el contenido del primer objeto del arreglo.

• **Estructura del Objeto de Proyecto:** Cada objeto dentro del arreglo “**proyects**” incluye varias propiedades que describen detalles importantes de un proyecto.



# Geoservicios

En esta parte encontramos los servicios web que utiliza este geoportal, nuestro proyecto de los geoservicios tiene el nombre de “GeoservicesPage.jsx”, para iniciar la estructura de la clase vamos a ver cada una de las partes.

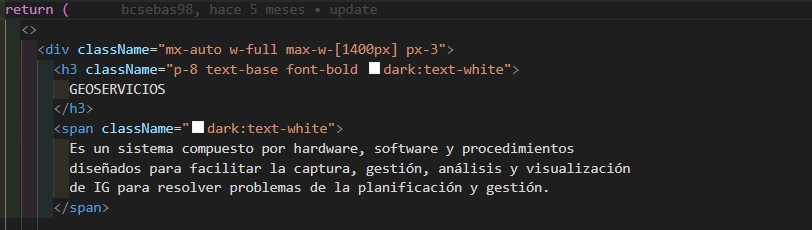
### Descripción

Este fragmento de código describe una sección de una interfaz de usuario diseñada para la plataforma que trabaja con información geográfica (IG). La sección se titula "GEOSERVICIOS" y proporciona una breve descripción de lo que implica un sistema de geoservicios. A continuación, se ofrece una guía general para documentar esta sección de código: **Descripción General**

• **Propósito y Contenido:** La sección está destinada a informar al usuario acerca de los geoservicios.

### Elementos del Código

* **Encabezado h3:** El encabezado <h3> destaca el título de la sección, "GEOSERVICIOS", utilizando clases de Tailwind CSS para estilización (p-8, text-base, font-bold, dark:text-white). Estas clases aplican un padding, establecen el tamaño del texto, el peso de la fuente, y ajustan el color del texto para modos claros y oscuros.
* **Descripción span:** El elemento <span> contiene la descripción de los geoservicios. Al igual que el encabezado, este elemento utiliza clases de Tailwind CSS para asegurar que el texto sea legible en temas oscuros (dark:text-white).



### Pestañas

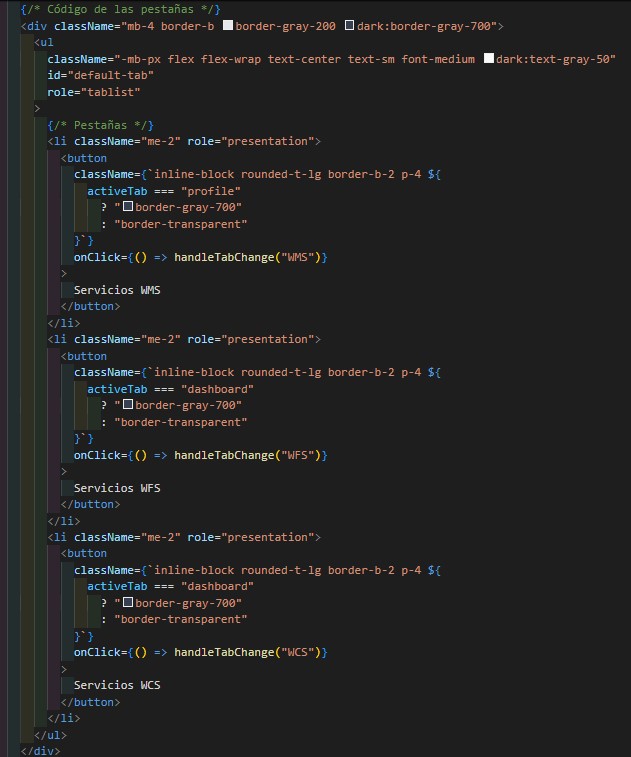
Este fragmento se utiliza para permitir a los usuarios cambiar entre diferentes vistas o secciones de contenido dentro de la misma página web. En este caso, está configurado para alternar entre diferentes tipos de servicios relacionados con sistemas de información geográfica: WMS (Web Map Service), WFS (Web Feature Service), y WCS (Web Coverage Service). A continuación, se detalla una guía general para documentar este código:

### Estructura y Funcionamiento

* **Contenedor Principal:** Se usa un “**div**” con clases de Tailwind CSS para estilos de borde y margen, actuando como contenedor del componente de pestañas.
* **Lista de Pestañas:** Las pestañas están implementadas como elementos “**li**” dentro de un elemento “**ul**”, facilitando una estructura semántica accesible y clara. Cada pestaña es un botón que, al hacer clic, activa un cambio de la vista mostrada al usuario.
* **Estilos Condicionales:** Se utilizan plantillas literales y expresiones ternarias para aplicar estilos condicionales a las pestañas. Los estilos cambian dependiendo de cuál pestaña está activa (activeTab), lo que indica visualmente al usuario cuál sección está viendo actualmente.
* **Manejador de Eventos:** Cada botón de pestaña tiene un evento onClick que invoca la función handleTabChange, pasando el identificador del servicio correspondiente (WMS, WFS, WCS). Esta función es responsable de cambiar el estado del componente y actualizar la vista activa.

### Consideraciones de Implementación

• **Flexibilidad:** Sugerir la posibilidad de extender el componente para incluir más pestañas o servicios, modificando la estructura de la lista y las funciones correspondientes.



### Contenido de las Pestañas

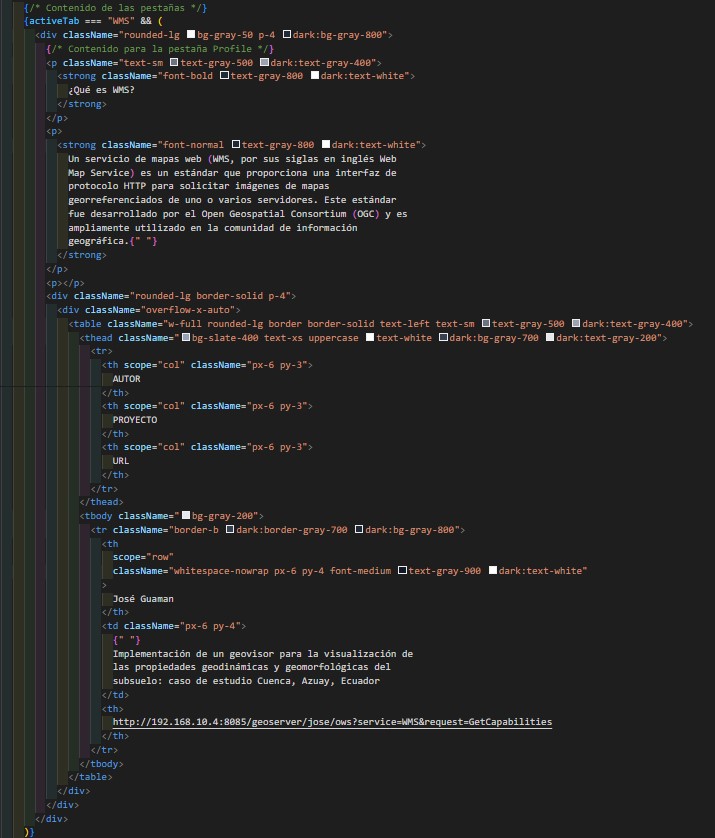
Esta sección específica pertenece a la pestaña “**Profile**” de una página web y se centra en proporcionar información sobre el “**Servicio de Mapas Web**”, junto con un ejemplo específico de proyecto que utiliza WMS. A continuación, se detalla la estructura y el propósito de este código:

### Estructura General

* **Descripción del WMS:** Se inicia con un párrafo breve que introduce qué es un WMS.
* **Tabla de Ejemplo:** A continuación, se presenta una tabla con estilo personalizado que contiene tres columnas: AUTOR, PROYECTO y URL. Esta tabla muestra un ejemplo de proyecto que utiliza WMS, incluyendo el nombre del autor (José Guaman), el nombre del proyecto, y una URL que parece apunta un servidor GeoServer configurado para ofrecer servicios WMS.

### Elementos de Código

* **Uso de clases CSS para estilización:** El código hace un uso intensivo de clases de Tailwind CSS para aplicar estilos, como colores de texto (text-gray-500, dark:text-gray-400), fondos (bg-slate-400, dark:bg-gray-700), bordes (border-solid, dark:border-gray-700), y otros para espaciado, redondeado de bordes, etc.
* **Elementos React:** El fragmento está construido utilizando JSX, una extensión de sintaxis para React que permite escribir estructuras similares al HTML en el código JavaScript.

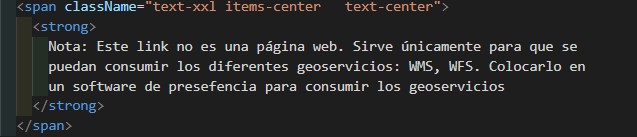


### Advertencia o Nota

Este fragmento es una advertencia o nota destinada a los usuarios, aclarando la función específica de un enlace proporcionado en el contexto de los geoservicios. A continuación, se detalla una guía para documentar esta sección de código:

### Descripción General

• **Propósito de la Nota:** La nota informa a los usuarios que el enlace proporcionado no conduce a una página web tradicional. En cambio, su propósito es ser utilizado dentro de un software específico para acceder a geoservicios como WMS y WFS. Esto es importante para evitar confusiones entre los usuarios que podrían esperar que el enlace abra una página web en su navegador.

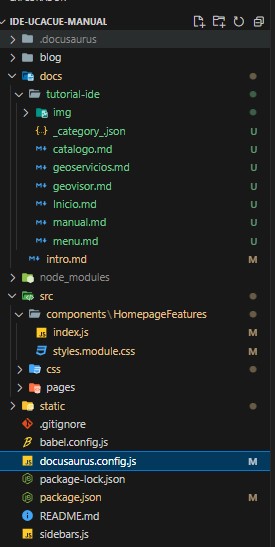


# Manual

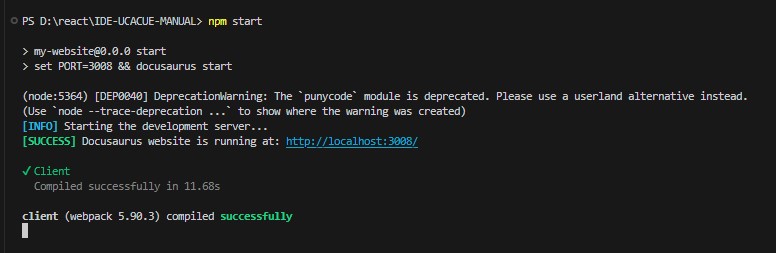
El manual se realizó utilizando Docusaurus que es una herramienta dedicada a la construcción, despliegue y mantenimiento de sitios web de documentación de código abierto. Está construida sobre React, lo que simplifica el proceso de escritura de la documentación al permitir a los usuarios escribir sus documentos en Markdown, que luego se convierten automáticamente en un sitio web estructurado.

Nuestro proyecto del manual tiene el nombre de “IDE-UCACUE-MANUAL”, para empezar, veremos la estructura de directorio que nos brinda:

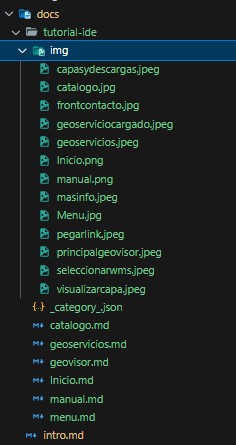
* **docs:** donde pondremos nuestros archivos Markdown para la documentación. o **src:** contiene los archivos fuente para el sitio, incluyendo páginas personalizadas.
* **docusaurus.config.js:** el archivo de configuración principal para el sitio, donde podemos definir el título, el tagline, el URL del proyecto, y más.



Como por ahora estamos desplegando de manera local, debemos irnos a la carpeta principal donde este nuestro “IDE-UCACUE-MANUAL”, abrimos el terminal y ejecutamos el comando *npm start* donde ya tenemos preconfigurado que corra en el puerto 3008.



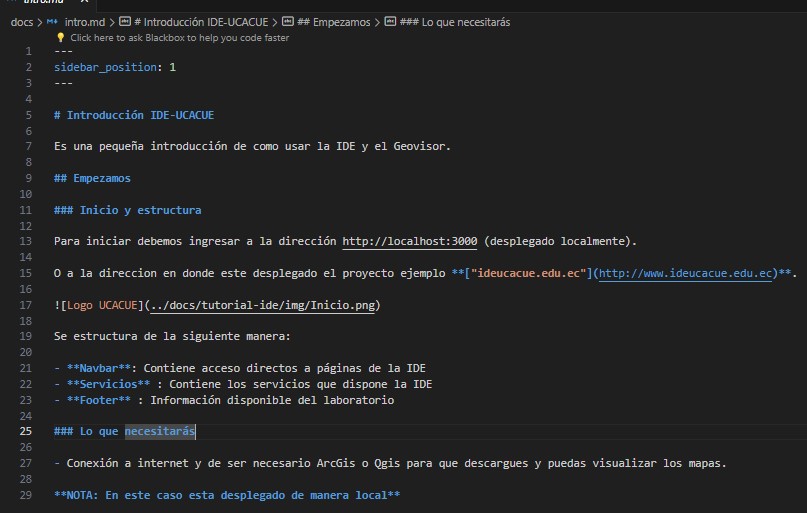
En esta carpeta de docs se encuentran todos los archivos de nuestra documentación como lo mencionamos antes:



Por cada componente que tenemos en la IDE, existe un archivo Markdown en el que explica el funcionamiento de este. Explicaremos en breve cada uno de ellos:

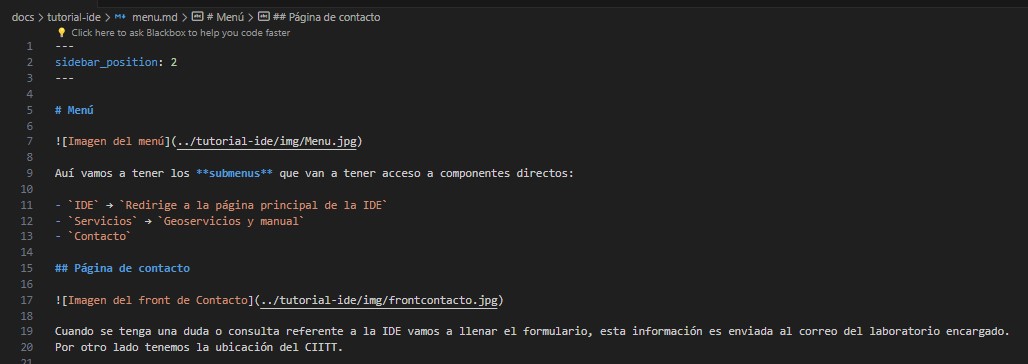
### 1) Inicio

Para editar la dirección IP en la que se publica el contenido, simplemente reemplace la dirección actual con la nueva, el inicio está en la primera posición del sidebar. Si desea modificar las imágenes, estas deben ser agregadas en la siguiente ruta: ../docs/tutorial-ide/img/nombreimagen.png. Asegúrese de incluir el nombre correcto del archivo de la imagen que desea agregar.



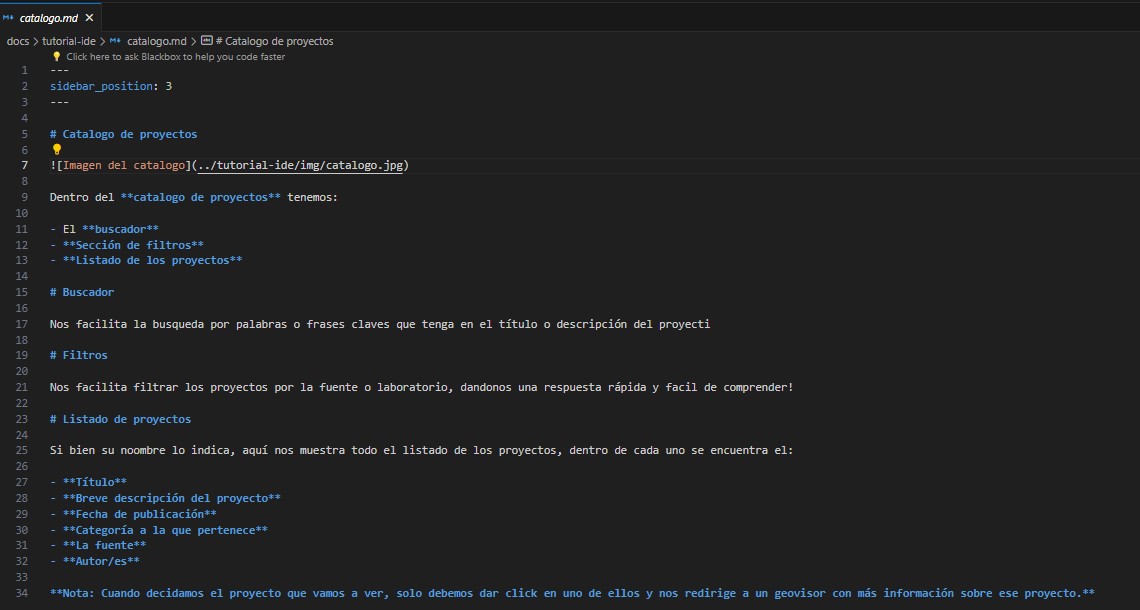
### 2) Menú

Como el inicio las imágenes se ubican en la ruta: ../docs/tutorial-ide/img/nombreimagen.png. Aquí se detalla submenús para una navegación directa hacia áreas específicas: el submenú "IDE" facilita el acceso a la página de inicio del IDE, "Servicios" conduce a una sección que ofrece geoservicios y documentación técnica, y "Contacto" abre una página dedicada a la comunicación con los usuarios. La sección de contacto, además de mostrar un formulario de consulta cuyo propósito es canalizar preguntas y comentarios hacia el correo electrónico del laboratorio a cargo, proporciona información de localización física del CIITT, permitiendo consultas en persona. Este enfoque estructural y de navegación busca optimizar la experiencia del usuario al proporcionar un acceso claro y directo a los recursos y servicios ofrecidos por el IDE.



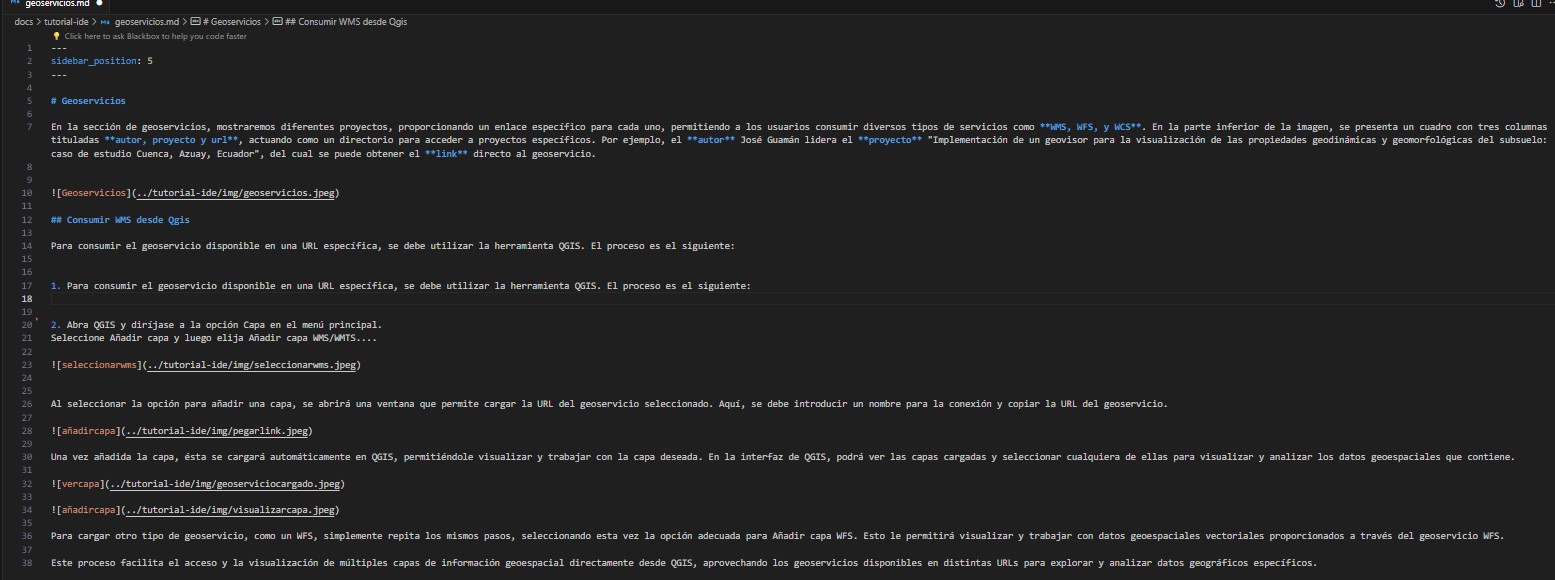
### 3) Catalogo

Se encuentra configurado para ocupar la tercera posición en el panel lateral, se presenta el "Catálogo de proyectos", acompañado de una imagen representativa. Este catálogo incluye un buscador para localizar proyectos por palabras clave, una sección de filtros que permite refinar la búsqueda según la fuente o laboratorio, y un listado completo de proyectos. Cada entrada en el listado detalla el título, una breve descripción, la fecha de publicación, la categoría, la fuente, y los autores del proyecto. Al seleccionar un proyecto, el usuario es redirigido a un geovisor que ofrece información detallada sobre el mismo, facilitando la navegación y el acceso a los datos de interés de manera eficiente y estructurada.



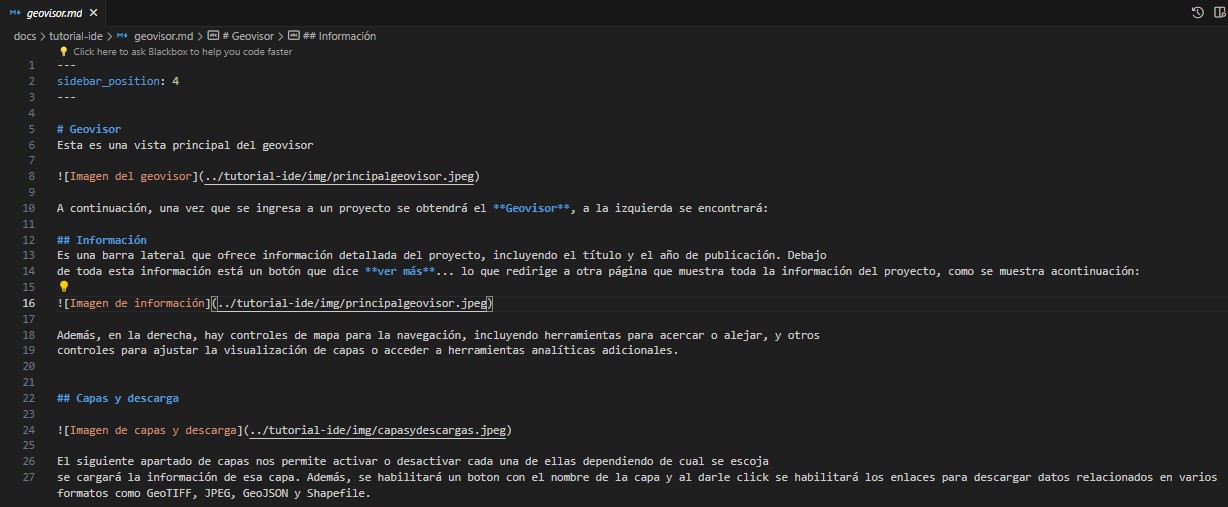
### 4) Geoservicios

Este componente está configurado para ser el quinto elemento en el panel lateral, detalla la sección "Geoservicios" con una guía técnica sobre cómo acceder y usar servicios geoespaciales como WMS, WFS y WCS a través de QGIS. Se inicia con una introducción a los geoservicios ofrecidos, seguida de un ejemplo práctico que incluye una imagen con un cuadro de autor, proyecto y URL para facilitar la referencia. Luego, se ofrece un tutorial paso a paso ilustrado para consumir un geoservicio WMS en QGIS, desde la apertura de la aplicación hasta la visualización de la capa geoespacial. Este enfoque instruccional proporciona a los usuarios un método claro y directo para integrar datos geoespaciales en sus proyectos de QGIS, demostrando el proceso a través de imágenes explicativas que acompañan cada paso.



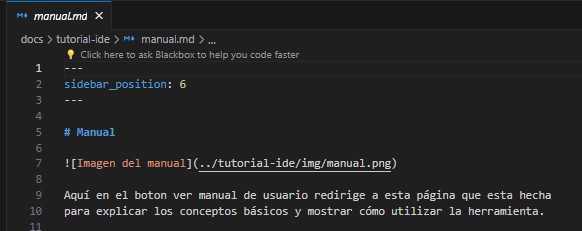
# 5) Geovisor

En la cuarta posición del panel lateral, el documento de Markdown presenta la sección "Geovisor", describiendo sus principales características y funcionalidades. Inicia con una visión general y una imagen del geovisor, seguido de detalles sobre la interfaz, que incluye una barra lateral con información del proyecto y controles de mapa para la navegación. La sección "Capas y descarga" ilustra cómo activar capas específicas y acceder a opciones de descarga de datos en formatos como GeoTIFF, JPEG, GeoJSON y Shapefile. Este resumen ofrece una guía concisa para el usuario sobre cómo navegar y utilizar el geovisor, resaltando sus capacidades de visualización y descarga de datos geoespaciales.



### 6) Manual

En la sexta y última posición del panel lateral, el documento de Markdown introduce la sección "Manual", complementada con una imagen que ilustra el acceso al mismo.



## Geovisor

Empezamos a describir la estructura y funcionalidad del componente ‘JoseMapsPage’ que también lo conoceremos como geovisor, las funcionalidades principales son visualización de un panel lateral con opciones para activar/desactivar capas de datos geográficos, descarga de datos geoespaciales en diferentes formatos (GeoTIFF, JPEG, GeoJson, Shapefile) y la interacción con un mapa geográfico para visualizar las capas de datos seleccionados.

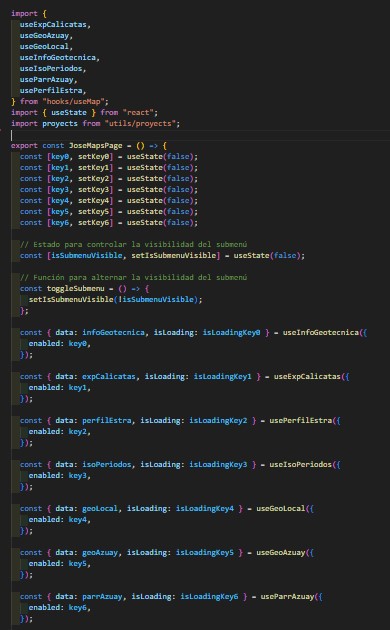
**Componentes importados:**

* **ChoseLayerItem:** Componente para cada item de la lista de capas de datos. Permite al usuario activar/desactivar la visualización de la capa en el mapa.
* **GeoViewerJose:** Componente encargado de mostrar el mapa y las capas de datos activas.
* **Link:** Componente de react-router-dom utilizado para la navegación entre componentes sin recargar la página.



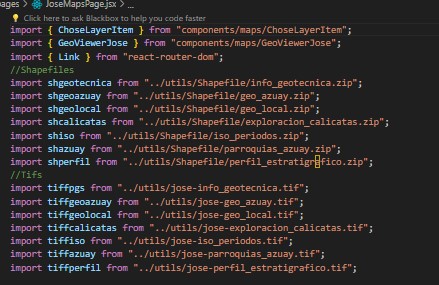
### Hooks

* **useState**: Se utiliza para manejar el estado de la visibilidad de las capas de datos y del submenú.
* **useExpCalicatas, useGeoAzuay, useGeoLocal, useInfoGeotecnica, useIsoPeriodos, useParrAzuay, usePerfilEstra:** Hooks personalizados que probablemente se encargan de cargar y administrar los datos para cada tipo de capa geográfica, incluyendo el manejo de estados de carga.



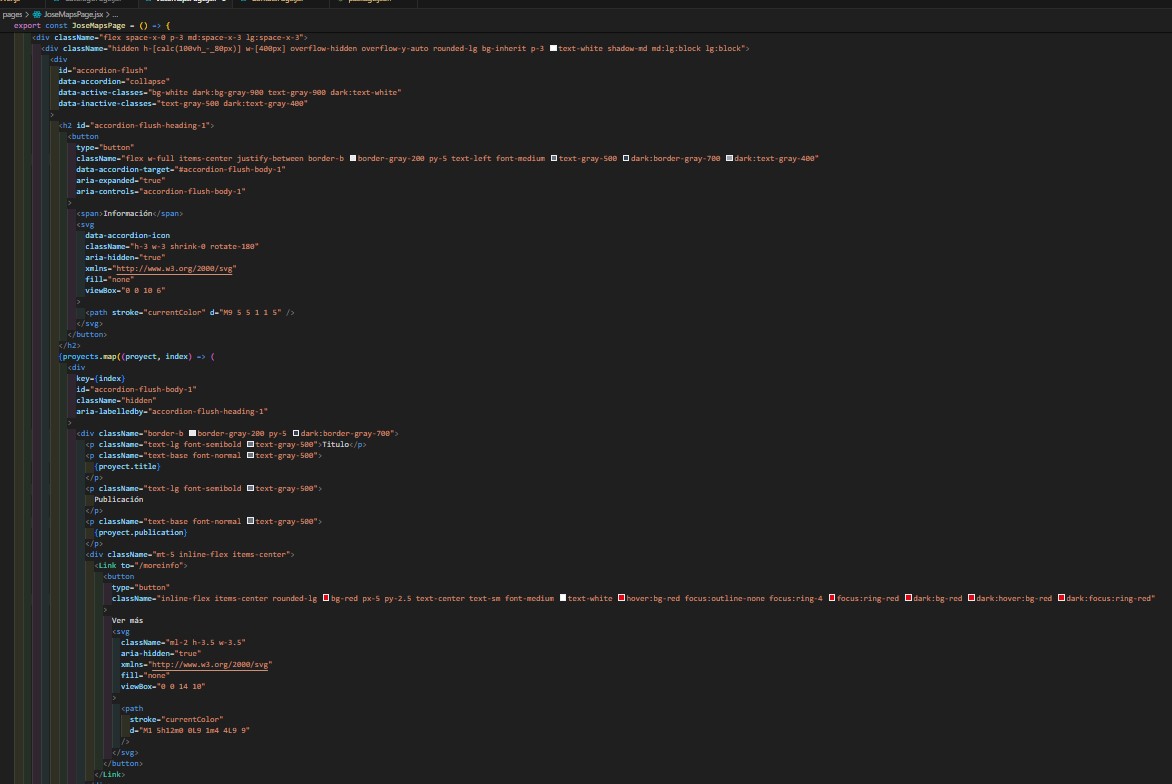
### Importaciones

• Diversos archivos ZIP y TIFF: Representan los datos geoespaciales que se pueden visualizar en el mapa y descargar por el usuario.



### Funcionalidad de la UI

• El componente organiza su interfaz en dos partes principales: un panel lateral y un visor de mapas. El panel lateral contiene una lista desplegable de proyectos y capas de datos, cada una con opciones para activar/desactivar la visualización en el mapa y descargar los datos. El visor de mapas muestra las capas activas seleccionadas por el usuario.



### Manejo de Estados y Datos

• El componente utiliza estados para controlar la activación de las capas y la visibilidad del submenú. Los hooks personalizados probablemente hacen peticiones a una API o a un servidor de mapas para obtener los datos geoespaciales correspondientes a cada capa activada.

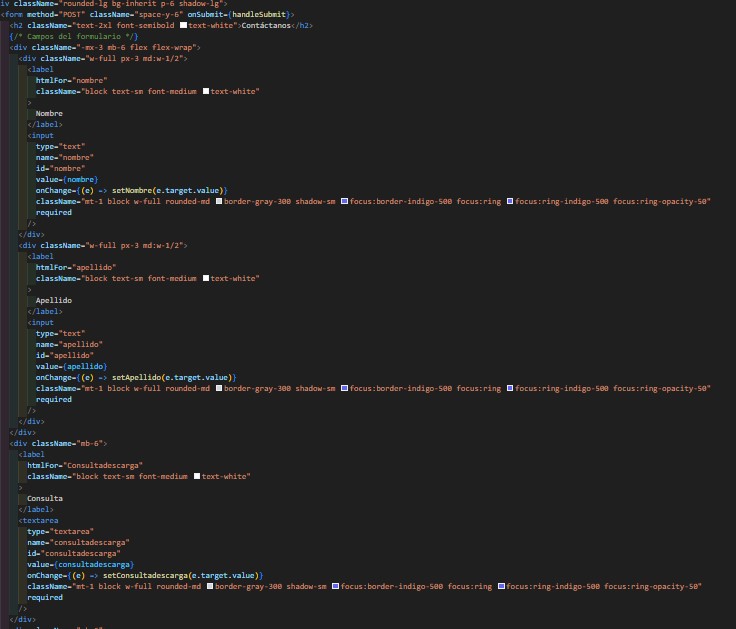


# Contactos

### Estructura del Componente

El componente se divide en dos secciones principales:

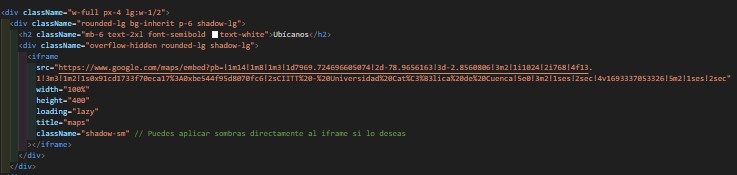
* **Formulario de Contacto:** Permite al usuario introducir sus datos personales, consulta, y detalles de afiliación.







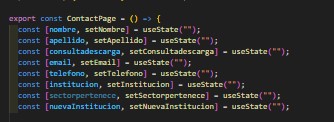
* **Ubicación:** Presenta un mapa de Google Maps embebido que muestra la ubicación relevante para el usuario.



### Estados

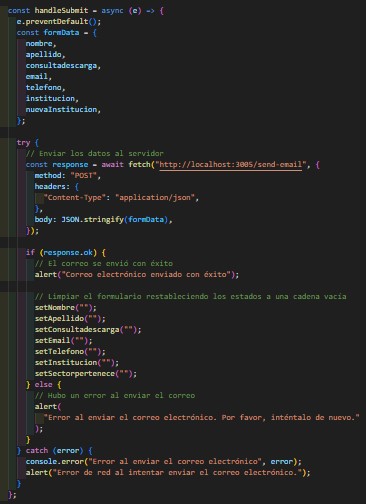
El componente maneja varios estados locales con useState para almacenar la información introducida por el usuario:

* **nombre:** Almacena el nombre del usuario.
* **apellido:** Almacena el apellido del usuario.
* **consultadescarga:** Contiene la consulta o mensaje que el usuario desea enviar.
* **email:** Guarda el correo electrónico del usuario.
* **telefono:** Almacena el número de teléfono del usuario.
* **institucion:** Guarda la afiliación institucional del usuario.
* **sectorpertenece:** Este estado no se utiliza en el código proporcionado y podría ser removido o implementado en futuras extensiones del componente.
* **nuevaInstitucion:** En caso de que el usuario seleccione "Otra" institución, este estado almacena el nombre de esa nueva institución.



### Funcionalidades Principales

* Envío de Datos: Al enviar el formulario, se recopilan los datos del estado y se envían a un servidor o API especificado, en este caso, a http://localhost:3005/send-email, usando fetch con método POST.
* Validación y Limpieza de Formulario: El formulario realiza validaciones básicas usando el atributo required de HTML5 en los campos y limpia los campos después de un envío exitoso.



### Estilos y Layout

* El componente utiliza estilos definidos en "../assets/css/home.css", lo que implica que su apariencia está dictada por las clases CSS definidas en ese archivo.
* Se emplean clases de Tailwind CSS para el layout y diseño del formulario, como rounded-lg, shadowlg, text-white, entre otras, para darle una apariencia moderna y responsive.

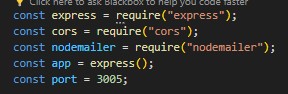


### • Server.js

Este servidor se encarga de recibir solicitudes HTTP POST con información de contacto y consultas de usuarios para luego enviar esta información a través de un correo electrónico.

### Configuración del Entorno

* **Express:** Marco de trabajo para aplicaciones web en Node.js que facilita la creación de servidores HTTP.
* **CORS (Cross-Origin Resource Sharing):** Módulo para habilitar CORS, permitiendo que el servidor acepte solicitudes de dominios cruzados.
* **Nodemailer**: Módulo para el envío de correos electrónicos desde Node.js.



### Instalación de Dependencias

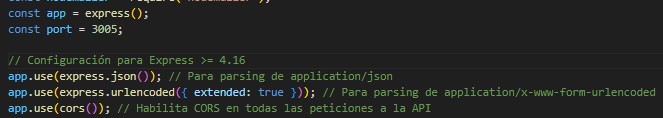
Para instalar las dependencias necesarias (express, cors, nodemailer), se debe ejecutar el siguiente comando en el terminal:

• *npm install express cors nodemailer*



### Configuración del Servidor

* **Puerto:** El servidor se configura para escuchar en el puerto 3005.
* **Middleware:** 
  + **express.json():** Permite al servidor manejar solicitudes que tengan un cuerpo en formato JSON.
  + **express.urlencoded({ extended: true }):** Habilita el procesamiento de cuerpos de solicitudes que usen codificación URL.
  + **cors():** Aplica CORS a todas las rutas del servidor, permitiendo solicitudes de cualquier origen.



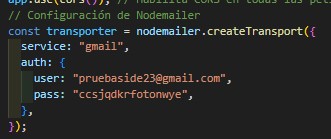
### Configuración de Nodemailer

Se configura un transporte de Nodemailer utilizando el servicio de correo electrónico de Gmail. Credenciales del gmail:

Correo: pruebaside23@gmail.com

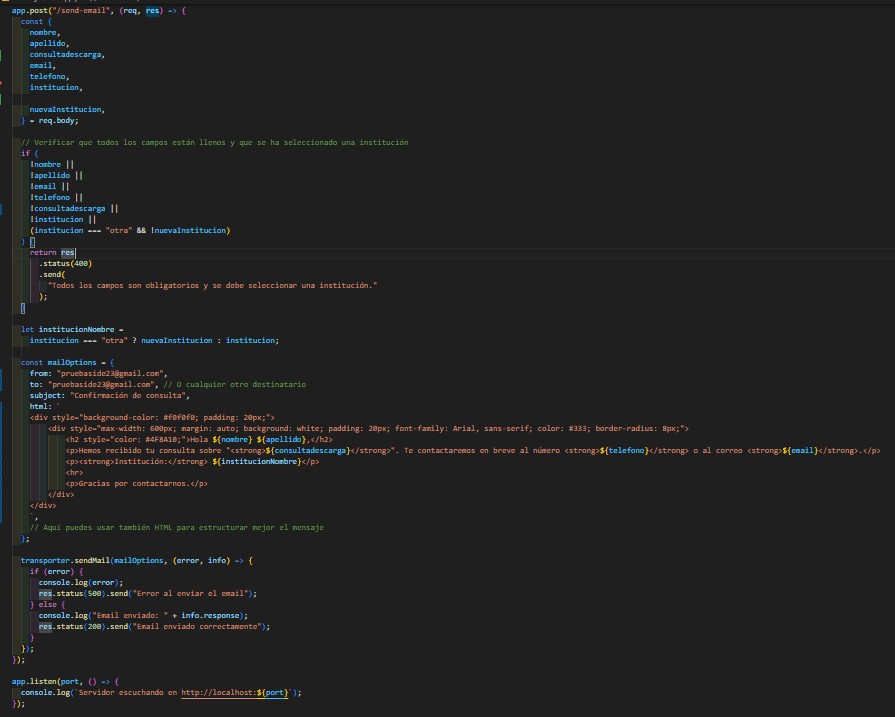
Contraseña: ciitt-server23

Es necesario proporcionar un usuario y una contraseña (en este caso, se usa una contraseña de aplicación generada para mayor seguridad).



### Ruta del Servidor

* **POST /send-email:** Esta ruta maneja las solicitudes POST para enviar correos electrónicos. Recibe datos de un formulario que incluye nombre, apellido, consulta, correo electrónico, teléfono, institución y, si corresponde, el nombre de una nueva institución.
* **Validación:** Se verifica que todos los campos requeridos estén presentes. Si falta alguno, se retorna un error HTTP 400.
* **Envío de Correo:** Si todos los campos están correctos, se procede a enviar un correo electrónico usando Nodemailer con los detalles proporcionados. En caso de error en el envío, se retorna un error HTTP 500.
* **Respuesta:** Se envía una respuesta indicando si el correo se envió correctamente o si ocurrió un error.



### Ejecución del Servidor

Para iniciar el servidor, se ejecuta el siguiente comando:

• *node server.js*

