Documentacion WEB

1

Generado por Doxygen 1.12.0

1. Biometría y Medio Ambiente - Web

Este proyecto es una plataforma web diseñada para el monitoreo de datos biométricos y ambientales. Permite la visualización y análisis de estos datos en tiempo real a través de una interfaz web amigable. Está construida utilizando tecnologías modernas como PHP para el backend, JavaScript para el frontend y Docker para la gestión de contenedores.

1.1. Tecnologías Utilizadas

- **Backend:** PHP
- **Frontend:** JavaScript, HTML, CSS
- **Contenedores:** Docker, Docker Compose
- **Gestor de dependencias:** npm
- **Base de datos:** MySQL

1.2. Instalación y Configuración

1.2.1. Prerrequisitos

1. **Docker**: Asegúrate de tener Docker instalado en tu sistema. `Instalar Docker`.
2. **Node.js y npm**: Instala `Node.js` y npm para gestionar las dependencias del frontend.

1.2.2. Instalación

1. **Clona el repositorio**:
   ```
   git clone https://github.com/SentoMarcos/Biometr-a-y-Medio-Ambiente-Docker-Web-DB.git
   ```
2. **Accede al directorio del proyecto**:
   ```
   cd Biometr-a-y-Medio-Ambiente-Docker-Web-DB
   ```
3. **Instala las dependencias del frontend**: Ejecuta el siguiente comando para instalar las dependencias necesarias:
   ```
   npm install
   ```
4. **Construye y ejecuta los contenedores con Docker**: Usa Docker Compose para construir y levantar todos los servicios:
   ```
   docker-compose up --build
   ```
5. **Accede a la aplicación**: Una vez que los contenedores estén en funcionamiento, puedes acceder a la aplicación web en tu navegador en la URL: `http://localhost:8000` (o el puerto especificado en el archivo `docker-compose.yml`).

1.2.3. Configuración de Variables de Entorno

Crea un archivo `.env` en la raíz del proyecto con las variables de entorno necesarias para la configuración de la base de datos y otros servicios. Aquí tienes un ejemplo:

```
DB_HOST=db
DB_PORT=3306
DB_USER=root
DB_PASSWORD=password
DB_NAME=biometria
```

Estas variables de entorno serán utilizadas por Docker para la configuración de los servicios internos.

1.3. Autores

- SentoMarcos

1.4. Proyectos Relacionados

- Biometría y Medio Ambiente - Android
- Biometría y Medio Ambiente - Arduino

2. Índice de clases

2.1. Lista de clases

Lista de clases, estructuras, uniones e interfaces con breves descripciones:

3. Índice de archivos

3.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

4. Documentación de clases

4.1. Referencia de la interface Botón

la última medición

Diagrama de colaboración de Botón:

4.1.1. Descripción detallada

la última medición

Interfaz para obtener la última medición registrada en la base de datos.

La documentación de esta interface está generada del siguiente archivo:

- medicionesgetter.js

4.2. Referencia de la interface Texto

la última medición

Diagrama de colaboración de Texto:

4.2.1. Descripción detallada

la última medición

Interfaz para mostrar la última medición registrada en la base de datos.

La documentación de esta interface está generada del siguiente archivo:

- medicionesgetter.js

5. Documentación de archivos

5.1. Referencia del archivo .env

5.2. .env

Ir a la documentación de este archivo.

```
00001 # Configuración de la base de datos
00002 DATABASE_HOST=mysqldb   # Cambia a 'localhost' si ejecutas fuera de Docker
00003 DATABASE_USER=root
00004 DATABASE_PASSWORD=123456
00005 DATABASE_NAME=mydb
00006 DATABASE_PORT=3306
00007
00008 # Configuración del servidor (opcional)
00009 SERVER_PORT=3000         # Cambia al puerto que desees usar para tu servidor
```

5.3. Referencia del archivo README.md

5.4. Referencia del archivo database.js

Variables

- import { createPool } from 'mysql2/promise'

  @fileoverview Archivo de configuración de la base de datos MySQL @description Este archivo contiene la configuración necesaria para establecer
- import dotenv from dotenv
- const pool

  @const pool @description Crear un pool de conexiones a la base de datos MySQL @type {Pool} @requires mysql2/promise @requires dotenv

5.4.1. Documentación de variables

dotenv

```
import dotenv from dotenv
```

Definición en la línea 8 del archivo database.js.

import

```
import { createPool } from 'mysql2/promise'
```

@fileoverview Archivo de configuración de la base de datos MySQL @description Este archivo contiene la configuración necesaria para establecer

Definición en la línea 6 del archivo database.js.

pool

```
export default pool
```

Valor inicial:

```
= createPool({
    host: process.env.DATABASE_HOST || 'localhost',
    user: process.env.DATABASE_USER || 'root',
    password: process.env.DATABASE_PASSWORD || '123456',
    database: process.env.DATABASE_NAME || 'mydb',
    port: process.env.DATABASE_PORT || 3306,
})
```

@const pool @description Crear un pool de conexiones a la base de datos MySQL @type {Pool} @requires mysql2/promise @requires dotenv

Parámetros

{Object}	host - Host de la base de datos
{Object}	user - Usuario de la base de datos
{Object}	password - Contraseña de la base de datos
{Object}	database - Nombre de la base de datos
{Object}	port - Puerto de la base de datos

Definición en la línea 25 del archivo database.js.

5.5. database.js

Ir a la documentación de este archivo.

```
00001
00006 import { createPool } from 'mysql2/promise';
00007 //ruta a env
00008 import dotenv from 'dotenv';
00009
00010
00011 dotenv.config();
00012
00025 const pool = createPool({
00026     host: process.env.DATABASE_HOST || 'localhost',
00027     user: process.env.DATABASE_USER || 'root',
00028     password: process.env.DATABASE_PASSWORD || '123456',
00029     database: process.env.DATABASE_NAME || 'mydb',
00030     port: process.env.DATABASE_PORT || 3306,
00031 });
00032
00033
00034 // Exportar el pool para ser utilizado en otros archivos
00035 export default pool;
```

5.6. Referencia del archivo PeticionesCTR.js

Controladores para gestionar las peticiones a la base de datos.

Variables

- import pool from Config database js
- import { obtenerUltimaMedicion, insertarMedicion } from '../../Logica/medicionesService.js'
- export const pingDB

  Realiza un ping a la base de datos.
- export const getUltimaPeticion

  Obtiene la última medición registrada en la base de datos.
- export const setMedicion

  Inserta una nueva medición en la base de datos.

5.6.1. Descripción detallada

Controladores para gestionar las peticiones a la base de datos.

Este archivo contiene los controladores para gestionar las peticiones a la base de datos.

Definición en el archivo PeticionesCTR.js.

5.6.2. Documentación de variables

getUltimaPeticion

```
export const getUltimaPeticion
```

Valor inicial:

```
= async (req, res) => {
    try {
        const ultimaMedicion = await obtenerUltimaMedicion();
        res.json(ultimaMedicion);
    } catch (error) {
        console.error('Error al obtener la última medición:', error.message);
        res.status(500).send('Error en la base de datos');
    }
}
```

Obtiene la última medición registrada en la base de datos.

Esta función obtiene la última medición registrada en la base de datos.

Parámetros

{Object}	req La solicitud HTTP.
{Object}	res La respuesta HTTP.

Definición en la línea 36 del archivo PeticionesCTR.js.

import

```
import { obtenerUltimaMedicion, insertarMedicion } from '../../Logica/medicionesService.js'
```

Definición en la línea 7 del archivo PeticionesCTR.js.

js

```
import pool from Config database js
```

Definición en la línea 6 del archivo PeticionesCTR.js.

pingDB

```
export const pingDB
```

Valor inicial:

```
= async (req, res) => {
    try {
        const [result] = await pool.query('SELECT NOW()');
        res.json(result);
    } catch (error) {
        console.error('Error al hacer ping a la base de datos:', error.message);
        res.status(500).send('Error en la base de datos');
    }
}
```

Realiza un ping a la base de datos.

Esta función realiza una consulta a la base de datos para comprobar si está operativa.

Parámetros

{Object}	req La solicitud HTTP.
{Object}	res La respuesta HTTP.

Definición en la línea 17 del archivo PeticionesCTR.js.

setMedicion

```
export const setMedicion
```

Valor inicial:

```
= async (req, res) => {
    try {
        const { Lugar, Gas, Valor } = req.body;
        if (!Lugar || !Gas || !Valor) {
            return res.status(400).send('Faltan datos obligatorios');
        }
        await insertarMedicion(Lugar, Gas, Valor);
        res.status(201).send('Medición insertada correctamente');
    } catch (error) {
        console.error('Error al insertar la medición:', error.message);
        res.status(500).send('Error en el servidor');
    }
}
```

Inserta una nueva medición en la base de datos.

Parámetros

req	
res	

Devuelve

{Promise<∗>}

Definición en la línea 53 del archivo PeticionesCTR.js.

5.7. PeticionesCTR.js

Ir a la documentación de este archivo.

```
00001
00006 import pool from '../Config/database.js';
00007 import { obtenerUltimaMedicion, insertarMedicion } from '../../Logica/medicionesService.js';
00008
00017 export const pingDB = async (req, res) => {
00018     try {
00019         const [result] = await pool.query('SELECT NOW()');
00020         res.json(result);
00021     } catch (error) {
00022         console.error('Error al hacer ping a la base de datos:', error.message);
00023         res.status(500).send('Error en la base de datos');
00024     }
00025 };
00026
00036 export const getUltimaPeticion = async (req, res) => {
00037     try {
00038         const ultimaMedicion = await obtenerUltimaMedicion();
00039         res.json(ultimaMedicion);
00040     } catch (error) {
00041         console.error('Error al obtener la última medición:', error.message);
00042         res.status(500).send('Error en la base de datos');
00043     }
00044 };
00045
00053 export const setMedicion = async (req, res) => {
00054     try {
00055         const { Lugar, Gas, Valor } = req.body;
00056         if (!Lugar || !Gas || !Valor) {
00057             return res.status(400).send('Faltan datos obligatorios');
00058         }
00059         await insertarMedicion(Lugar, Gas, Valor);
00060         res.status(201).send('Medición insertada correctamente');
00061     } catch (error) {
00062         console.error('Error al insertar la medición:', error.message);
00063         res.status(500).send('Error en el servidor');
00064     }
00065 };
00066
00067 //borrar ultima medicion
00068
00069 export const borrarUltimaMedicion = async (req, res) => {
00070     try {
00071         await borrarUltimaMedicion();
00072         res.status(204).send('Medición borrada correctamente');
00073     } catch (error) {
00074         console.error('Error al borrar la medición:', error.message);
00075         res.status(500).send('Error en el servidor');
00076     }
00077 }
```

5.8. Referencia del archivo PeticionesRutas.js

Rutas para gestionar las peticiones a la base de datos.

Variables

- import { Router } from 'express'
- const router = Router()

5.8.1. Descripción detallada

Rutas para gestionar las peticiones a la base de datos.

Este archivo contiene las rutas para gestionar las peticiones a la base de datos.

Definición en el archivo PeticionesRutas.js.

5.8.2. Documentación de variables

import

```
import { Router } from 'express'
```

Definición en la línea 6 del archivo PeticionesRutas.js.

router

```
export default router = Router()
```

Definición en la línea 9 del archivo PeticionesRutas.js.

5.9. PeticionesRutas.js

Ir a la documentación de este archivo.

```
00001
00006 import { Router } from 'express';
00007 import { pingDB, getUltimaPeticion, setMedicion, borrarUltimaMedicion } from
      '../CTR/PeticionesCTR.js';
00008
00009 const router = Router();
00010
00019 router.get('/ping', pingDB);
00020
00029 router.get('/ultima-medicion', getUltimaPeticion);
00030
00040 router.post('/insertar', setMedicion);
00041
00050 router.delete('/borrar', borrarUltimaMedicion);
00051
00052 export default router;
```

5.10. Referencia del archivo index.js

Archivo principal para iniciar el servidor.

Variables

- import express from express
- import cors from cors
- import path from path
- import { fileURLToPath } from 'url'
- import PeticionesRutas from api Rutas PeticionesRutas js
- const app = express()
- const __filename = fileURLToPath(import.meta.url)
- const __dirname = path.dirname(__filename)

5.10.1. Descripción detallada

Archivo principal para iniciar el servidor.

Definición en el archivo index.js.

5.10.2. Documentación de variables

__dirname

```
const __dirname = path.dirname(__filename)
```

Definición en la línea 21 del archivo index.js.

__filename

```
const __filename = fileURLToPath(import.meta.url)
```

Definición en la línea 20 del archivo index.js.

app

```
const app = express()
```

Definición en la línea 13 del archivo index.js.

cors

```
import cors from cors
```

Definición en la línea 6 del archivo index.js.

express

```
import express from express
```

Definición en la línea 5 del archivo index.js.

import

```
import { fileURLToPath } from 'url'
```

Definición en la línea 8 del archivo index.js.

js

```
import PeticionesRutas from api Rutas PeticionesRutas js
```

Definición en la línea 9 del archivo index.js.

path

```
import path from path
```

Definición en la línea 7 del archivo index.js.

5.11. index.js

Ir a la documentación de este archivo.

```
00001
00005 import express from 'express';
00006 import cors from 'cors';
00007 import path from 'path';
00008 import { fileURLToPath } from 'url';
00009 import PeticionesRutas from './api/Rutas/PeticionesRutas.js';
00010
00011
00012 // Crear la aplicación express
00013 const app = express();
00014
00015 // Configuraciones de middleware
00016 app.use(express.json());
00017 app.use(cors());
00018
00019 // Configuración para servir archivos estáticos
00020 const __filename = fileURLToPath(import.meta.url);
00021 const __dirname = path.dirname(__filename);
00022 app.use(express.static(path.join(__dirname, 'public')));
00023
00024 // Rutas
00025 app.use('/api', PeticionesRutas); // Usar rutas bajo el prefijo /api
00026
00030 app.get('/', (req, res) => {
00031     res.sendFile(path.join(__dirname, 'public', 'index.html'));
00032 });
00033
00034 // Iniciar el servidor
00035 app.listen(process.env.PORT || 3000, () => {
00036     console.log(`Server running at http://localhost:${process.env.PORT || 3000}/`);
00037 });
```

5.12. Referencia del archivo medicionesService.js

Variables

- import pool from api Config database js
- export const obtenerUltimaMedicion

  Obtiene la última medición registrada en la base de datos.
- export const insertarMedicion

  Inserta una nueva medición en la base de datos.
- export const borrarUltimaMedicion

  Elimina la última medición registrada en la base de datos.

5.12.1. Documentación de variables

borrarUltimaMedicion

```
export const borrarUltimaMedicion
```

Valor inicial:

```
= async () => {
  const [result] = await pool.query('DELETE FROM medidas ORDER BY fecha DESC LIMIT 1');
  return result;
}
```

Elimina la última medición registrada en la base de datos.

Esta función elimina la última fila de la tabla de medidas, ordenada por fecha de manera descendente.

Devuelve

{Object} El resultado de la operación de eliminación, incluyendo información sobre las filas afectadas.

Definición en la línea 53 del archivo medicionesService.js.

insertarMedicion

```
export const insertarMedicion
```

Valor inicial:

```
= async (Lugar, Gas, Valor) => {
  const fecha = new Date();
  const [result] = await pool.query(
      'INSERT INTO medidas (fecha, Lugar, Gas, Valor) VALUES (?, ?, ?, ?)',
      [fecha, Lugar, Gas, Valor]
  );
  return result;
}
```

Inserta una nueva medición en la base de datos.

Esta función inserta una nueva fila en la tabla de medidas con los valores proporcionados para lugar, tipo de gas y valor de medición. La fecha se genera automáticamente en el momento de la inserción.

Parámetros

{String}	Lugar El lugar donde se realizó la medición.
{String}	Gas El tipo de gas medido.
{Number}	Valor El valor medido del gas.

Devuelve

{Object} El resultado de la operación, que incluye información sobre las filas afectadas.

Definición en la línea 35 del archivo medicionesService.js.

js

```
import pool from api Config database js
```

Definición en la línea 2 del archivo medicionesService.js.

obtenerUltimaMedicion

```
export const obtenerUltimaMedicion
```

Valor inicial:

```
= async () => {
  const [result] = await pool.query('SELECT * FROM medidas ORDER BY fecha DESC LIMIT 1');
  if (result.length === 0) {
    throw new Error('No se encontraron mediciones');
  } else if (result.length > 1) {
    throw new Error('Error al obtener la última medición');
  }
  return result[0];
}
```

Obtiene la última medición registrada en la base de datos.

Esta función realiza una consulta a la base de datos para obtener la última medición almacenada, ordenando por fecha de manera descendente.

Excepciones

Error	Si no se encuentran mediciones o si se obtienen múltiples resultados.

Devuelve

{Object} Un objeto que representa la última medición, incluyendo los campos `Lugar`, `Gas`, `Valor` y `fecha`.

Definición en la línea 14 del archivo medicionesService.js.

5.13. medicionesService.js

Ir a la documentación de este archivo.

```
00001 // Logica/medicionesService.js
00002 import pool from '../api/Config/database.js';
00003
00014 export const obtenerUltimaMedicion = async () => {
00015   const [result] = await pool.query('SELECT * FROM medidas ORDER BY fecha DESC LIMIT 1');
00016   if (result.length === 0) {
00017     throw new Error('No se encontraron mediciones');
00018   } else if (result.length > 1) {
00019     throw new Error('Error al obtener la última medición');
00020   }
00021   return result[0];
00022 };
00023
00035 export const insertarMedicion = async (Lugar, Gas, Valor) => {
00036   const fecha = new Date();
00037   const [result] = await pool.query(
00038       'INSERT INTO medidas (fecha, Lugar, Gas, Valor) VALUES (?, ?, ?, ?)',
00039       [fecha, Lugar, Gas, Valor]
00040   );
00041   return result;
00042 };
00043
00053 export const borrarUltimaMedicion = async () => {
00054   const [result] = await pool.query('DELETE FROM medidas ORDER BY fecha DESC LIMIT 1');
00055   return result;
00056 };
```

5.14. Referencia del archivo medicionesgetter.js

Funciones para obtener mediciones de la base de datos y mostrarlas en la página web.

Variables

- const getUltimaMedicion

  @function getUltimaMedicion
- const boton = document.getElementById('btn-ultima-medicion')
- const texto = document.getElementById('texto-medicion')

5.14.1. Descripción detallada

Funciones para obtener mediciones de la base de datos y mostrarlas en la página web.

Este archivo contiene funciones para obtener mediciones de la base de datos y mostrarlas en la página web.

Definición en el archivo medicionesgetter.js.

5.14.2. Documentación de variables

boton

```
const boton = document.getElementById('btn-ultima-medicion')
```

Definición en la línea 33 del archivo medicionesgetter.js.

getUltimaMedicion

```
const getUltimaMedicion
```

Valor inicial:

```
= async () => {
    try {

        const response = await fetch('/api/ultima-medicion');
        if (!response.ok) {
            throw new Error('Error al obtener la última medición');
        }
        return await response.json();
    } catch (error) {
        console.error('Error al obtener la última medición:', error);
        return null;
    }
}
```

@function getUltimaMedicion

Obtiene la última medición registrada en la base de datos.

Realiza una petición GET a la ruta "/api/ultima-medicion" para obtener la última medición registrada en la base de datos.

Devuelve

La última medición registrada en la base de datos.

Excepciones

Error	si hay un error al obtener la última medición.

Definición en la línea 14 del archivo medicionesgetter.js.

texto

```
const texto = document.getElementById('texto-medicion')
```

Definición en la línea 40 del archivo medicionesgetter.js.

5.15. medicionesgetter.js

Ir a la documentación de este archivo.

```
00001
00014 const getUltimaMedicion = async () => {
00015     try {
00016         // Asegúrate de que la ruta aquí coincida con el backend (usando el prefijo '/api' si
      corresponde)
00017         const response = await fetch('/api/ultima-medicion');
00018         if (!response.ok) {
00019             throw new Error('Error al obtener la última medición');
00020         }
00021         return await response.json();
00022     } catch (error) {
00023         console.error('Error al obtener la última medición:', error);
00024         return null;
00025     }
00026 };
00027
00032 // Obtener el botón del DOM
00033 const boton = document.getElementById('btn-ultima-medicion');
00034
00039 // Obtener el elemento del DOM donde se mostrará la medición
00040 const texto = document.getElementById('texto-medicion');
00041
00047 boton.addEventListener('click', async () => {
00048     // Llama a la función para obtener la última medición
00049     const medicion = await getUltimaMedicion();
00050
00051     // Si se obtiene la medición, actualiza el contenido de 'texto'
00052     if (medicion) {
00053         // Verifica que las propiedades 'fecha', 'Lugar', 'Gas' y 'Valor' sean correctas según tu API
00054         texto.textContent = `Última medición: Fecha: ${medicion.fecha}, Lugar: ${medicion.Lugar}, Gas:
      ${medicion.Gas}, Valor: ${medicion.Valor}`;
00055     } else {
00056         // Muestra un mensaje de error si no se pudo obtener la medición
00057         texto.textContent = 'Error al obtener la última medición';
00058     }
00059 });
```

5.16. Referencia del archivo medidasService.test.js

Variables

- import { expect } from 'chai'
- import dotenv from dotenv

5.16.1. Documentación de variables

dotenv

```
import dotenv from dotenv
```

Definición en la línea 10 del archivo medidasService.test.js.

import

```
import { expect } from 'chai'
```

Definición en la línea 9 del archivo medidasService.test.js.

5.17. medidasService.test.js

Ir a la documentación de este archivo.

```
00001
00009 import { expect } from 'chai';
00010 import dotenv from 'dotenv';
00011
00012 // Cargar las variables de entorno desde .env.test
00013 dotenv.config({ path: '.env.test' });
00014
00015 import { obtenerUltimaMedicion, insertarMedicion, borrarUltimaMedicion } from
      '../src/Logica/medicionesService.js';
00016
00020 describe('Servicios de Medición', () => {
00021
00028     it('Debe obtener la última medición correctamente', async () => {
00029         const medicion = await obtenerUltimaMedicion();
00030         expect(medicion).to.have.property('Lugar');
00031         expect(medicion).to.have.property('Gas');
00032         expect(medicion).to.have.property('Valor');
00033     });
00034
00041     it('Debe insertar una nueva medición correctamente', async () => {
00042         const result = await insertarMedicion('Laboratorio', 'CO2', 23.5);
00043         expect(result.affectedRows).to.equal(1);
00044     });
00045
00052     it('Debe borrar la última medición', async () => {
00053         await insertarMedicion('Laboratorio', 'O3', 30.0);
00054         const result = await borrarUltimaMedicion();
00055         expect(result.affectedRows).to.equal(1);
00056         const medicionBorrada = await obtenerUltimaMedicion();
00057         expect(medicionBorrada).to.have.property('Lugar', 'Laboratorio');
00058     });
00059 });
```

Índice alfabético